PREDICTION OF PARAMETERS TO AVOID VEHICLE ROLL OVER USING NEURAL NETWORKS.

By

Helen Jean Cunningham, BE (Hons) *Mech.*

Submitted in fulfilment of the requirements for the Degree of

Engineering (Civil & Mechanical)

Master of Engineering Science

at the

University of Tasmania (June 2002)

STATEMENT OF ORIGINALITY AND AUTHORITY OF ACCESS.

This thesis contains no material that has been accepted for a degree or diploma by the University of Tasmania or any other institution, except by way of background information and has been duly acknowledged in the thesis, and to the best of the author's knowledge and belief no material has been previously published or written by another person except where due acknowledgment is made in the text of the thesis.

This thesis contains confidential information and is not to be disclosed or made available for loan or copy without the express permission of the University of Tasmania (i). Once released the thesis may be made available for loan and limited copying in accordance with the Copyright Act 1968.

(i) Inquiries should be directed to the Research and Development Office.

Signed: H. Cunningham

Dated this nineteenth day of June 2002

ABSTRACT

There is a need for reliable automotive performance. While automotive engineers are highly trained mechanical engineers, there is a requirement to keep abreast of the emerging technologies such as neural networks or fast-converging algorithms. Any significant or radical change comes about through multi-disciplinary interaction. Emerging technologies such as evolutionary algorithms, neural networks and fuzzy logic are constantly applied to more diverse technological applications.

From automotive industry point of view, continual attempts are made to build models to avoid vehicle roll over. While highly advanced automotive manufacturers are carrying out such research, very little or no results are available in the public domain.

In this thesis, critical parameters responsible for vehicle roll over will be identified and predicted. As part of the model verification, a hardware comprising of a Formula SAE race-car, sensory technology and instrumentation will be developed. This thesis highlights successful application of roll-over parameters namely longitudinal velocity, v, and vehicle roll angle, θ_r. This prediction is seen as a step towards identifying on-line warning systems for roll over detection and subsequent control systems to avoid roll over.

ACKNOWLEDGEMENTS

The work presented in this thesis was conducted as part of the 'intelligent car' research team here at the University of Tasmania and in collaboration with the University of Stralsund, Germany.

I would like to acknowledge the kind support and guidance of my supervisor Dr Vishy Karri during every stage of the preparation of this thesis. His vision has been integral to maintaining momentum throughout the project.

I would like to thank the workshop staff for their assistance and tolerance during the manufacturing stage of the project, in particular Bernard Chenery and John McCulloch for their patient technical support in the development of a working electrical system for the test vehicle.

I would also like to thank my fellow team members for their tireless efforts: David Butler and Garth Heron for their work on the preliminary design and construction, Nick Jones and Robert Neben for their work on the measurement system, Nick Dwyer for the construction of the nose cone and Cranston Polson for his work on the engine.

On a personal note, I would like to thank my family for their endless support and encouragement and my flatmates Lucy Guastalegname and Michelle Harper for their patience, support and friendship throughout.

TABLE OF CONTENTS

TABLE OF FIGURES

Chapter 1 – Introduction

The introduction of the automobile ownership to the general public by Henry Ford in the early part of the last century has created a moving lifestyle for millions of people across the globe. The advantages of this lifestyle include greater freedom and flexibility of travel for all car owners, massive employment in the industries which design, produce and maintain these machines as well as the industries that fuel them. Cars have for many years provided users with an economic, flexible and often-fun form of transport to get them from one point to another for whatever reason. Sadly it seems few gains come without a loss. The disadvantage of mass individually controlled transport is greater scope for human error, and increased risk for all road users.

In the US in 1960, the Dr. William Haddon became the first director of the National Highway Safety Bureau (NHSB). Haddon, a public health physician recognised that standard public health methods and epidemiology could be used to study and prevent motor vehicle and other injuries. He examined interactions between host (human), agent (motor vehicle) and environment (highway), before during and after the incident concluding that each phase could be tackled to minimise the injuries.[1] In this thesis the agent or motor vehicle is considered and a further step toward preventing accidents proposed for this phase.

In the majority of studies done, human error is blamed for a significant proportion of motor vehicle related injuries (host). Some common causes of this error are excessive alcohol consumption, fatigue, driver inattention and driver inexperience. Much has already been done from the fields of public health and education to minimise the likelihood of drivers getting behind the wheel in these conditions. A great deal has also been done in highway improvement to make safer roads and to encourage drivers to obey speed limits (environment). The third prong to this attack lies clearly with the motor vehicle itself (the agent).

1.1 Need for sensor control in automobiles

Sensor control is a reasonably old concept. As with most new technology, Science-fiction writers have been suggesting it for years. Only recently, however, have the computational power and support technologies begun to mature to the required level. The idea of sensor control is to use a number of sensors throughout the car to measure and collect instantaneous data pertaining to the vehicle's current state of motion. This information may then be used to control aspects of the vehicle's dynamic behaviour including lateral sliding and vehicle roll over. Depending on the level of control this may mean; a split second emergency warning, suggested course of action or removal of control from the driver to the sensor control system. Each level of control naturally requires an additional increase in the reliability and accuracy of the system. (A false alarm is less dangerous than is an incorrect driver override strategy.)

Sensor control will allow for safer, more comfortable driving for all motorists using public highways. It will allow greater control over fuel economy and eventually will allow drivers to elicit the maximum life from their vehicles and vehicle components. This will lead to greater reliability and investment return for the automobile owner.

1.2 Concept of intelligent car used for traffic control, auto navigation and prevention of vehicle roll over.

Intelligence and artificial intelligence has many definitions. For the purpose of this thesis we will define the concept of the intelligent vehicle as a vehicle possessing the ability to make driving decisions based on its environment and previous events in similar environments. This vehicle will make decisions based on the data it obtains from its sensors and from trends built up from the data it has previously received. In this respect the vehicle is expected to 'learn from its experience'. This dynamic experience then becomes vital in critical situations where driver inexperience can prove fatal such as high-speed sliding and potential vehicle roll over.

The intelligent highway is a concept intended to maximise the flow rate of traffic along highways while increasing passenger safety. The ideal of this concept includes a navigation component whereby the most efficient route between locations is always

taken. Development of intelligent highway technologies aims to reduce traffic congestion and while maximising economy and safety. Many systems proposed thus far have required the installation of expensive cameras and sensors into current highway systems. A feasible solution proposes the use of sensors and transmitters already being built into many new cars to send information back and forth to a central body.[2] As such it seems the 'sensitive car' may be a critical component in the development of the intelligent highway as well as dynamic vehicle safety.

1.3 Need for reliable estimation of dynamic performance

Central to the technology for controlling vehicle dynamics is, not only the output from the sensors, but also the interpretation of this output. For real-time critical driving control, it is necessary to have split second information about what is currently happening to the dynamics of the vehicle. This is the role of reliable dynamic performance estimation. The control systems based on the sensors are useless for altering the outcome an event if the information arrives too late for the control system to activate. Since sensors are not yet able to sense into the future a reliable estimation of what is currently occurring is needed. This estimation must be based on the precise sensory outputs, which are coordinated to indicate the true 'state' the vehicle is in. Such an estimate allows the setting of control limits that would then allow impending difficulty to be controlled.

1.4 Current state of affairs

While the majority of the major automotive manufacturers is investigating sensor control, and in many cases, producing models featuring localised forms of this control, progress tends to centre on individual systems rather than a more global approach. The advantage of a global or comprehensive approach becomes apparent in reduced cost for additional features.[3]

As many of the control features require the same inputs and utilise the same controls, (eg. ASS and ABS both use individual wheel speed and brake pressure in their control systems) it is possible to attain both systems for little more than the initial cost of just

one system. The clear advantage is economic for both the consumer and the manufacturer

1.5 Difference between local control and comprehensive control

As previously stated there is a clear distinction between what might be termed local control and global or comprehensive of vehicle dynamics. A discussion and some examples of the two systems follow.

Local control refers to control systems that focus on one aspect of the automotive system only. While many of these systems may appear on the one vehicle the control loop for each relies only on the components it controls and does not coordinate with any other system on board. There are already many examples of this type of control on the modern car. To follow are three of the more common control systems.

1.5.1 Cruise control

Cruise control allows the maintenance of a set speed, chosen by the driver. The driver sets the required speed and from then on no input is required from the accelerator pedal to maintain the speed. Within the control loop, the engine RPM is varied to maintain the vehicle speed obtained from a wheel speed sensor. The simple feedback loop means that the speed will remain constant relative to the ground regardless of any inclines or declines that may be encountered during the journey. In each case the control loop will raise or lower the engine RPM to match the wheel speed with the value set by the driver. The system may be immediately over-ridden by a touch on the brake pedal or accelerator that allows for emergency manoeuvres should they be required. Minor problems with the system may include slow response to decline speed changes, a problem in the age of speed cameras, and a reduction in the need for driver concentration which may lead to inattentiveness or boredom whilst driving.

Adaptive cruise control is the next step in this line of development where the distance to the vehicle in front is set to allow constant speed following as is often the case in congested traffic.[4] Such a system requires forward 'seeing' sensors to determine the distance or time to the next vehicle as well as an advanced control system to allow for quick braking and acceleration.

1.5.2 ABS

ABS is the German acronym used for Anti-lock Braking. These systems are now wide spread in the automotive industry and are becoming a standard on new vehicles. In heavy braking or braking under low friction conditions the wheels undergoing the braking action may be caused to lock up or skid due to the force on the wheel from the brake being greater than the force of the wheel in contact with the road. As it is known that the coefficient of sliding friction is somewhat lower than that of static friction, the braking force from wheels that are skidding (sliding) will be less than that of wheels that continue to roll (stationary at point of contact). In addition to this a rolling tyre can exert a lateral force on the road whereas a sliding tyre cannot, hence the ability to steer is lost when sliding occurs (this is discussed in greater detail in chapter 2). To prevent these effects, the anti-lock brake system senses when the wheels are stationary ie have 'locked up' and are skidding and forces the brake pressure to release to force the wheels to allow them to roll again.

This system is constantly active on the vehicle and to some extent takes absolute control of the brake pressure away from the driver. It was for this reason somewhat reluctantly accepted after its introduction. As may be expected drivers are hesitant to relinquish control to a system until it has been well proven. It has over the years proven itself to be a reliable and valuable asset for safer driving.

1.5.3 ASS

ASS or Anti-Skid Steering is a control strategy reduces the angle of steering to that which the road can take without sliding.[5] This is much the same principal as ABS, which reduces the brake forces to that which the road can take without skidding. The value of such a system is supported by the great difficulty encountered by drivers other than professional drivers when trying to control a skidding vehicle. Clearly avoiding the skid or potential roll over situation to begin with would be preferable. The difficulty encountered with ASS that was encountered also with ABS is proving that the system will always perform better than a driver without the system performs. It seems that this may be quite difficult to prove beyond the consumer's doubt.

1.5.4 Comprehensive control

Comprehensive control is the integration of a number of local controls into one system. In this form of control the local units are able to assist for the best overall performance of the vehicle. Comprehensive control may mean using individual wheel braking to control the maximum allowable steering angle whilst avoiding vehicle roll over. It may mean using on-line engine control to maintain speed in cruise control or control of headway whilst still allowing the driver to dictate the movements of the car. The major objective is to allow the driver to drive up to the vehicle's limits without crossing them as this would create an unsafe environment for the driver and other road users. As a comprehensive system, the best means available may be utilised to avoid incident as opposed to the modular, local approaches that allow intervention by one system only. With current computing power now capable of real time decision-making and computational speed continuing to increase rapidly, the technological support for such a system is becoming available.

A number of manufacturers have working systems that begin to emulate this concept at least to some degree. At this time a complete working system of comprehensive control does not appear to exist in the public domain. It is certainly not yet available to the consumer.

1.6 Need for comprehensive control to avoid vehicle roll over

Although much has been done in this area to date, a primary need still exists in the automotive industry to continue to reduce the number of deaths and injuries through motor vehicle accidents. Many localised systems save lives either through direct application eg. ABS, or through reduction of driver fatigue which is a known cause of accidents eg. Cruise control. To prevent accidents of a catastrophic nature, such as vehicle roll over, requires an integrated comprehensive approach. Accidents will continue to occur, and as the number of vehicles using the public roads increases so must the standard of safety equipment with which a vehicle is fitted. A great many local systems are available but integration is required to prevent redundancy between the systems. The increase in efficient use of resources available will quickly bring down the price of such systems to the consumer; thus making them accessible to a

broader cross-section of the community. The safer cars become and the greater the number of safe cars the lower the likelihood of accidents and the lower the cost of accidents to the community.

1.7 Need for intelligent tools

A complex problem such as predicting the parameters that contribute to vehicle roll over is ideally suited to the application of intelligent tools, by this is meant the use of artificial neural networks. These networks are specifically suited to the numerical estimation of complex non-linear relationships such as will be found in the study of vehicle dynamics. However, the use of intelligent tools such as artificial neural networks implies a direct need for training and training data. The backbone of the neural network is its ability to produce the same results as the particular system without specific knowledge of that system. These tools have been proven on many manufacturing systems already and the applications continue to expand.

Similarly an intelligent system can be of use in other than critical driving conditions through the control of driving economy. Such a system would allow the driver to be aware of bad habits that are costing money in the form of excess fuel or excessively worn parts. It may also allow for control to be implemented maintaining a set value of efficiency. However, the training data must be of high quality specific to its purpose. The computer acronym GIGO is pertinent in this case, garbage in garbage out means that for a system to behave comprehensively it must be trained comprehensively. This requires the use of a database built up from extensive track trials of the highest quality standard. Accurate comprehensive data will yield accurate comprehensive results.

1.8 What has been done to date

The majority of work completed to date focuses on one or other aspect of sensor control. Many systems have been developed encompassing control of only one of many systems of the automobile. Overlap between the systems has been largely ignored and the result is great room for centralised use of technology. As with the majority of competitive industries 'the edge' in automobile manufacturing often lies

in the evolution of technology, for this reason much of the information relating to sensor control tools and technology remains in the private domain.

Finally, a reliable estimation of parameters responsible primarily for vehicle roll over, as a cause of accidents will be identified and predicted in this thesis.

Chapter 2 Vehicle Dynamics Control and Applied Intelligence

To estimate parameters responsible for roll over, an understanding of vehicle dynamics is imperative. While the use of neural networks reduces the need to completely understand the physical characteristics of a system when modelling it, it is still useful to have a conceptual understanding of the dynamics involved. This is to build the basis for decisions utilised in control systems.

2.1 Conceptual Vehicular Physics

Regardless of a vehicle's state of movement, a wide range of highly varied forces act on a vehicle in motion. One group of these forces occurs along the longitudinal axis; including propulsive force, aerodynamic drag and rolling resistance. Another group acts laterally, these include crosswinds and the centrifugal forces generated during cornering. These forces are transferred to the tyres and ultimately to the road surface by a number of transfer elements, these are

- The chassis (eg wind force),
- The steering (steering force),
- The engine,
- The transmission (propulsive force), and the
- Brake system (braking force).

A number of forces act upon the vehicle "from below", these include the lateral and longitudinal forces resulting from the gradient of the road and its transverse slope.[6]

When the forces mentioned reach extreme magnitudes, during say, major shifts in the vehicle's state of motion, they can become dangerous (eg skidding) which may result in an accident occurring. It is thus important when considering the dynamic handling response of a vehicle to consider the following parameters:

- Steering-wheel angle,
- Lateral acceleration,
- Linear acceleration/deceleration,
- Yaw rate,
- Float and roll angles

Additional data that will allow more precise definition of specific handling patterns:

- Longitudinal and transversal velocity (lateral velocity),
- Steering angles of front and rear wheels,
- Slip angles at all wheels,
- Torque applied at the steering wheel.[6]

Fundamental to the understanding of vehicle dynamics that cause roll over is a detailed knowledge of forces acting upon the vehicle. The most complex of these interactions is the force through the tyres. A discussion of tyre dynamics follows including friction factor, slip and overall vehicle dynamics and some applications already developed that deal with these factors.

The tire is the connecting element linking the vehicle and the road surface. The tire transfers propulsive, braking and lateral forces. This environment means the vehicle's load limits are defined by various physical factors.[6]

2.1.1 Vertical Tyre Force

The vertical tyre force is the downward force exerted at the contact patch between the tyre and the road. This force consists of the vehicle weight and its load distributed over the wheels. The road longitudinal gradient and lateral grade add extra components. Additional loads on the vehicle can increase or decrease the vertical tyre forces. In the case of traversing a curve, the effect would be to increase the load on the outer tyres and decrease the load on the inner tyres. The effect of additional load is to reshape the contact patch, which is not uniform as the tyre sidewalls take much of the load and prevent uniformity.

2.1.2 Longitudinal Force

When rolling along a road or other surface the wheel rotation rate is proportional to the wheel hub's linear velocity, if we consider for the time being that tyre rolling-resistance is ignored. This relationship is affected by external influences acting on the wheel, such as a brake force decelerating the wheel. The resulting interrelationship produces wheel slip.

2.1.3 Tyre Slip

Tyre slip corresponds to the difference between the theoretical and actual distances covered by the tyres. For example, if the circumference of a standard passenger car tyre were roughly 1.5 meters, then it would be logical to assume those ten rotations of the wheel translate into 15 meters of vehicle travel. In reality, the actual distance is shorter owing to the tyre slip.

Tyre slip originates from the tyre's inherent inflexibility. A driven wheel rolling along the road surface is subject to deformation while simultaneously flexing with varying intensity according to weather and road surface conditions. This leads to energy consumption and heats the tyre. Because the tyre's primary constituent is rubber, only a portion of this deformation energy is recovered as the tyre leaves its contact zone or patch.

Under braking and deceleration, as well as during acceleration – either from standing or rolling start – the level of force transfer depends upon the tyres' slip rates at the road surface. The relationship between slip and the tyre's coefficient of friction is basically the same whether accelerating or braking. Similarly the force transfer depends upon the tyres' slip rates at the road surface. The relationship between slip and the coefficient of friction is basically the same regardless of whether the vehicle is accelerating or decelerating. The vast majority of braking and acceleration processes take place at minimal slip rates within a stable range; here, increases in slip will be followed by a corresponding rise in available adhesion.

2.1.4 Friction Factor

Application of braking torque to the wheel generates a braking force, say F_B, between the tyre and the road surface. Under steady-state operation that is no wheel acceleration, this braking force is proportional to the braking torque. The relationship between the tyres' vertical contact force and the braking force that can be transmitted to the road is defined by the coefficient of static friction μ_{HF}.[6]

The coefficient of static friction, which is the maximum coefficient of adhesion, varies to reflect changes in such factors as vehicle speed, tyre condition and road-

surface. The static friction coefficient reflects the properties that materialise when tyre and road-surface materials meet, as well as all of the subsidiary influences that act on this combination. Actual figures are thus directly affected by road-surface condition.

2.1.5 Aquaplaning

A layer of rainwater on the road can cause the friction coefficient to dive towards zero as the vehicle is lifted from the tractive surface of the road. This phenomenon is known as "aquaplaning" and its distinctive feature is the loss of physical contact between tyre and road that occurs when a wedge of water forms to separate the two across the entire contact patch.

The tendency to aquaplane is defined by:

- The depth of the water on the road surface,
- The vehicle's speed,
- Tyre tread pattern and wear,
- The load pressing the tyre against the road.[6]

Wide tyres are particularly susceptible to aquaplaning. It is not possible to brake or steer an aquaplaning vehicle, as neither steering inputs nor braking force can be transferred to the road.

2.1.6 Friction – tyre slip.

The friction generated by a tyre is primarily determined by its longitudinal (rotational) slip. While vertical tyre force plays a subsidiary role, a roughly linear relationship exists between braking force and vertical tyre force during constant tyre slip.

Yet another force defining the friction is the wheel's slip angle (lateral slip). Whereas with constant tyre slip, the transfer of braking and motive forces decrease in response to higher wheel slip angles, increases in the wheel slip angle with constant braking and motive force will result in higher slip rates.

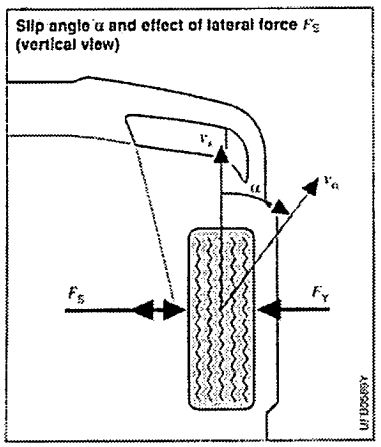


Figure 2.1-1 Wheel Slip Angle [6]

A freely rotating wheel reacts to application of lateral force with sideways movement at the hub. The ratio of lateral speed to longitudinal speed is termed lateral slip. The angle separating the resulting wheel-speed and the longitudinal speed is the slip angle α as shown in Figure 2.1-1. Under steady-state operation (without wheel acceleration) the axial force applied to the wheel through the axle as lateral force F_S in a state of equilibrium with the lateral forces exerted on the wheel through the road surface. The ratio of the lateral force transferred through the axle to the wheel's vertical tyre force F_N is the coefficient of lateral force μ_s.[6]

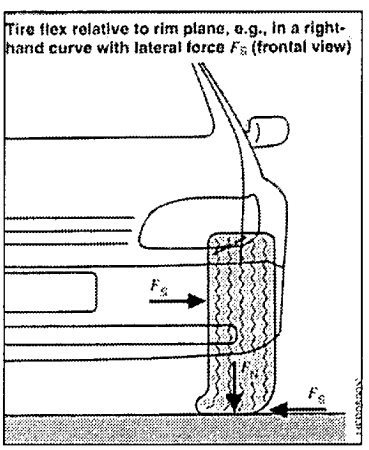


Figure 2.1-2 Ratio of Lateral Tyre Force to Vertical Tyre Force [6]

The relationship between the slip angle α and the coefficient of lateral force μ_s is non-linear and is defined by the slip-angle curve. The coefficient of lateral force μ_s contrasts with the coefficient of static friction μ_{HF} by exhibiting substantial sensitivity to the vertical tyre force F_N during acceleration and braking. This characteristic is of

special interest for the auto manufacturers' suspension designers in their attempts to improve the handling characteristics and prevent vehicle roll over by means of anti-roll bars. Very high lateral forces F_S induce substantial shifts in the position of the contact patch relative to the wheel rim and in doing so delay the build up of lateral forces as shown in Figure 2.1-2. This phenomenon has a substantial effect on the transition response (handling during the switch from one condition to another) that characterise vehicles when reacting to inputs at the steering wheel.

This affects the simplicity of estimation of velocities and accelerations dramatically. In the simple case of a vehicle travelling under high traction, estimates of velocities and accelerations can be made from the steering angle and wheel speed sensors. However, when the wheels begin to slide as previously mentioned the non-linearity of the relationship means that estimation of velocities and accelerations becomes increasingly complex.

2.1.7 Force Relationships

When lateral forces join the braking force acting upon a wheel rim, the road surface reacts by exerting two forces against the tyre, along both the braking and lateral axes. Providing the processes remain below a given physical threshold, all the forces acting upon the rotating wheel are effectively counterbalanced by opposite forces of equal magnitude from the road surface. Crossing this physical threshold upsets this state of equilibrium and results in a loss of vehicle stability.

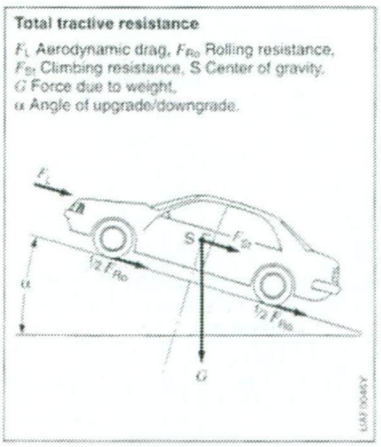


Figure 2.1-3 Total Tractive Resistances [6]

Total tractive resistance as shown in Figure 2.1-3 is the sum total of rolling, aerodynamic and climbing resistance. Overcoming this overall resistance entails applying sufficient tractive force to the driven wheels. The tractive force available at the driven wheels increases to reflect rises in such factors as available engine torque and the conversion ratio of the gearing between engine and wheels. It is inversely proportional to drive train losses. A proportion of the tractive force is needed to overcome total tractive resistance. Lower gearing in the form of numerically higher step-down conversion ratios is employed for graduated adaptation to the radical rise in tractive resistance encountered on uphill grades (multiple ratio transmission). The "surplus" by which tractive force exceeds tractive resistance accelerates the vehicle. If tractive resistance is higher than tractive force the vehicle will decelerate.

Rolling resistance originates from the deformation processes between wheel and road surface. It is the product of the force to weight and the coefficient of rolling resistance. Rolling resistance, in turn, is inversely proportional to tyre radius and the tire's degree of deformation (affected by such factors as tyre pressure). Rolling resistance also increases in response to higher loads and road speeds. Yet another factor is paving material; the coefficient of rolling resistance on asphalt is only approximately 25% of that on dirt roads.

During cornering, the rolling resistance is joined by cornering resistance, whose coefficient or magnitude is defined by such factors as vehicle speed, cornering radius, suspension geometry, tyre design, inflation pressures and the vehicle's cornering response (lateral acceleration at various slip angles).[6]

2.1.8 Aerodynamic Resistance

Aerodynamic resistance is determined based on a number of individual elements. These include barometric pressure, the vehicle's aerodynamic drag coefficient (as determined by its shape), the maximum vehicle cross-section and vehicle speed, taking headwind velocity into account.

Clearly, if all force directions and dynamic behaviour are to be found from first principles and used for prediction and control, it is complex.

Intense crosswinds shift vehicles from their initial paths of travel in a process that is especially pronounced at higher vehicle speeds and with unfavourable vehicle dimensions. On vehicles with unfavourable configurations, sudden wind impact of the kind encountered when a vehicle emerges from a tree-lined passage into open countryside can induce substantial lateral displacement and yaw angle shifts. When these phenomena manifest themselves before the driver has had time to react they can lead to driver error. Gusts of wind acting on the vehicle at an angle add a lateral component to the aerodynamic drag. This force, distributed across the entire surface area of the vehicle, can be considered as a single crosswind force applied at a "pressure point D" shown in Figure 2.1-4.

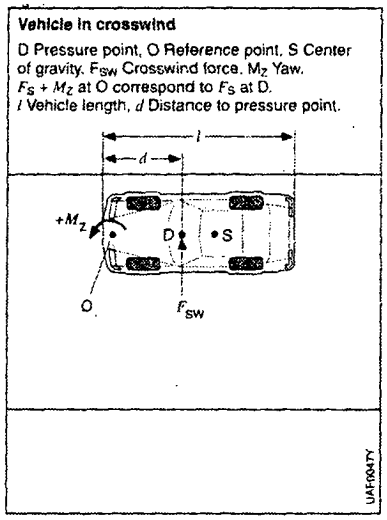


Figure 2.1-4 Vehicle in Crosswind

The precise location of this pressure point depends upon the shape of the body and the air current's angle of incidence although it is usually found on the forward half of the vehicle. On vehicles featuring a conventional "3-box" body configuration this focal point is relatively consistent, and also lies closer to the centre than on a hatchback, where the pressure point can wander in response to changes in the air flow's angle of attack. On the other hand, the centre of gravity S varies as a function of vehicle load. Generally selecting a reference point O in the middle of the vehicle's forward section facilitates applicable portrayals of crosswind effects (regardless of suspension position relative to the bodywork).

When crosswind force is defined using a reference point other than the pressure point, the crosswind force at the pressure point must be included as an additional factor: this is the yaw moment. The wheels' lateral guiding force (cornering forces) acts as a counterforce to the crosswind force. Along with the slip angle and load factor, the lateral guiding force generated by a pneumatic tyre depends upon its size and dimensions, inflation pressure and the friction characteristics of the road surface.
A pressure point location in the immediate vicinity of the vehicle's centre of gravity is desirable owing to its positive effects on directional stability under crosswind conditions. On vehicles with a natural tendency to oversteer a pressure point forward of the centre of gravity will minimise the tendency to wander from the original course. On understeering vehicles, the optimal pressure point location is immediately to the rear of the centre of gravity.[6]

2.1.9 Oversteer and Understeer

The wheel with its rubber tire must be rotating at an angle relative to its plane as a condition for lateral guiding forces (cornering forces) between wheel and road surface. This means that a slip angle must be present. Vehicles are described as having understeer when the slip angle of the front end increases more rapidly than the rear slip angle as lateral acceleration rises. The inverse condition (higher rear slip) is referred to as oversteer. Some vehicles display an intrinsic and invariable tendencies toward either oversteer or understeer, whatever the conditions. Others understeer at low rates of lateral acceleration before making a transition to oversteer as lateral-acceleration rises. Here, again, an inverted response pattern is also possible (initial oversteer and subsequent understeer).

2.1.10 Longitudinal Dynamics

The vital importance of stopping distances in road safety means that the distance travelled during deceleration is more significant than that travelled during acceleration. Accelerative and decelerative maxima are obtained with the tractive or braking forces acting on the vehicle wheels to hold them just below their traction limit (point of maximum adhesion). Real-world adhesion is lower because all wheels cannot uniformly exploit maximum adhesion under every accelerative (decelerative)

process. Electronic traction, braking and stability systems (TCS, ABS and ESP) rely on closed-loop control to maintain force transfer in the maximum range.[6]

2.1.11 Control – ESP

ESP stands for Electronic Stability Program and was developed by Robert Bosch GmbH. It is a vehicle stability system that relies on the vehicle's braking system as a tool for "steering" the vehicle. It is also known as VDC or vehicle dynamic control system in Mercedes vehicles. The desired trajectory of the vehicle is determined from the driver's inputs, steering wheel angle, engine drive torque as derived from the accelerator pedal position, and the brake pressure. The trajectory actually taken by the vehicle under these conditions will vary depending on the road friction. If the road is slippery, with a coefficient of friction, μ less than the nominal lateral acceleration, the vehicle will not follow the nominal motion and the radius of the turn will become larger than that of the nominal motion as shown in Figure 2.1-5.[7]

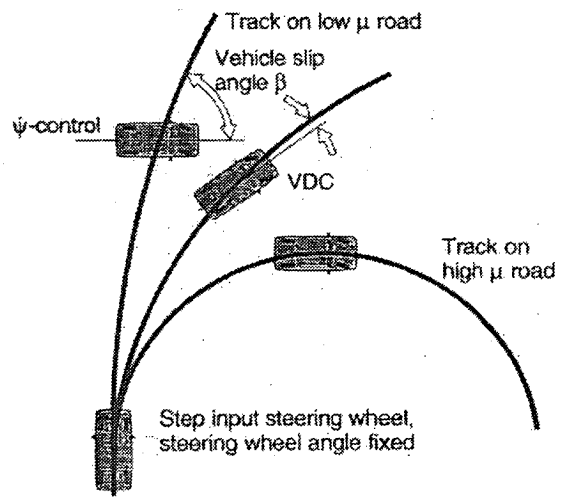


Figure 2.1-5 VDC Versus Yaw Rate Control [7]

One of the basic state variables that describe the lateral motion of the vehicle is its yaw rate. It would seem reasonable to design a control system that makes the yaw rate of the vehicle equal to the yaw rate of the nominal motion (yaw rate control). If this control is used on the slippery road, the lateral acceleration and the yaw rate will not correspond to each other as they do during the nominal motion. In fact the slip angle of the vehicle increases rapidly and the vehicle 'spins out'. Hence, both the yaw rate and the slip angle of the vehicle must be limited to values that correspond to the

coefficient of friction of the road. Thus in VDC both the yaw rate and the vehicle slip angle are taken as the nominal state variables and this as the controlled variables.[7]

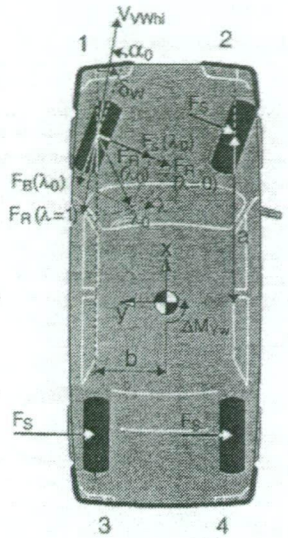


Figure 2.1-6 Rotation of tyre force.[7]

It is well known that both the longitudinal and lateral forces (F_L, F_S) on a tyre depend on the tyre slip λ, the slip angle α, and on the normal force on the tyre, F_N. The lateral force a tyre generates for a given slip angle decreases with increasing magnitude of the tyre slip. This property is used for the control of the lateral force and the yaw moment on the vehicle and, therefore, the tyre slip is used as the basic control variable of the control algorithm. For instance, if the slip of the left front tyre is increase by a small amount $\Delta\lambda$ from an initial value λ_o and if the tyre slip angle is α_o, then the yaw moment on the car is in a first approximation changed by the following amount:

$$\Delta M_{YW} = -\frac{\partial F_S}{\partial \lambda} \cdot \Delta\lambda (a \cdot \cos\delta_W - b \cdot \sin\delta_W) + \frac{\partial F_B}{\partial \lambda} \cdot \Delta\lambda (a \cdot \sin\delta_W + b \cdot \cos\delta_W)$$

Equation 2-1 [7]

Here, changes in the tyre normal force as a result of a change in the tyre longitudinal or lateral force are neglected, as are changes in the aligning torque on the tyre. Similarly, the lateral and the longitudinal forces on the vehicle will be changed by the following amounts:

$$\Delta F_x = -\frac{\partial F_S}{\partial \lambda} \cdot \Delta\lambda \cdot \sin\delta_W - \frac{\partial F_B}{\partial \lambda} \cdot \Delta\lambda \cdot \cos\delta_W \qquad \textbf{Equation 2-2 [7]}$$

$$\Delta F_y = -\frac{\partial F_S}{\partial \lambda} \cdot \Delta\lambda \cdot \cos\delta_W - \frac{\partial F_B}{\partial \lambda} \cdot \Delta\lambda \cdot \sin\delta_W \qquad \textbf{Equation 2-3 [7]}$$

These relations which can be derived for each wheel of the vehicle are extremely non-linear, since the derivatives of the forces are highly dependent on the operating point (λ_o,α_o) of the tyre.[7]

If the tyre slip is increased to the value λ_o, then the lateral force on the tyre is reduced to the value $F_S(\lambda_o)$. At the same time a brake force $F_B(\lambda_o)$ is generated. $F_R(\lambda_o)$ is now the resultant tyre force. At the limit of adhesion between the tyre and the road the absolute values of F_R (λ=0) and the $F_R(\lambda_o)$ are approximately equal. Clearly, increasing the tyre slip then means rotating the resultant tyre force and therefore changing the yaw moment, the lateral force, and the longitudinal force on the vehicle.

The rotation can be done at each tyre so that we can freely choose at which tyre the slip should be changed, and by which amount. Unfortunately, the changes in the longitudinal forces may lead to an undesired increase in the lateral deviation of the vehicle from the nominal path. A compromise result can be solved by optimal design methods. Special attention must be given to the robustness of the design since the operating point (λ_o,α_o) of the tyre is unknown (neither the tyre slip nor the slip angle is measured) and many related variables have estimated values only.[7]

Because the "discriminatory" control concept relies on two individual intervention strategies, the system has two options for "steering" the vehicle: it can brake selected wheels, known as selective braking or accelerate the driven wheels.[6] Although advanced, these tools do not identify or predict parameters responsible for vehicle roll over.

2.1.12 Control – ABS

Anti-lock braking or ABS has been around for at least 70 years, the first patent being issued in 1928 to Karl Wessel for a brake force controller that was never built. Robert

Bosch in 1936 and Fritz Osthaus in 1940 performed fundamental work. Fritz Ostwald patented it in 1936, during his undergraduate study. Heinz Leiber at Daimler –Benz brought about the first working system in 1964.[8]

The idea behind ABS is simple: the braking force coefficient and braking effectiveness are highest with the tyre at optimal brake slip. The controller modulates the brake pressure to keep the wheel in the optimal zone. The friction coefficient of a locked wheel is about 10% lower than optimum, depending on the surface. Even more important is the lateral, or sideways, force coefficient, since it decreases to only about 10% of its full value when the wheel locks.[8] This decrease in the lateral force can allow a vehicle to slide sideways out of a curve instead of maintaining its trajectory.

ABS systems do not necessarily exploit all available traction. In 2 channel systems only the wheel speed at one front wheel is sensed which can lead to lockup or under-braking of the other front wheel. In 3 channel systems and some 4-channel systems the rear wheels are low-value-tied, and the modulation is based on the wheel with the lower adhesion. Only the very best ABS systems have four wheel sensors and can use all adhesion at each wheel. But even then, it may be preferable to low-value-tie the rear wheels for better stability, as in Mercedes', since the overriding goal is stable vehicle dynamics.[8]

The goal of every ABS system is to provide minimum stopping distance and maximum directional control. Minimum stopping distance is achieved by maintaining the peak longitudinal force the tyre is capable of. Typically this value occurs at 5 to 15% longitudinal slip between the tyre and road. Some older systems have feedback of one vehicle state variable: lateral acceleration. Later systems make use of additional information available from a yaw velocity sensor, steering wheel angle (and velocity) as well as other pertinent parameters.[9] Maximum directional control means maintaining the ability to make the vehicle turn proportional using the steering wheel as in normal driving situations. This is maintained by not allowing the wheels to lock.

Unfortunately ABS has been found to have a number of drawbacks under certain circumstances. For example below speed of 50 kph, ABS systems have been found to drop to as low as 82% of the deceleration of a standard braking system average deceleration with locked, sliding wheels.[10] Areas in which ABS continues to be improved include the following: economy, making the systems available to even the lowest cost vehicles; improvement of available traction usage under complex real road conditions to extend the range of performance; and to improve the interface of vehicle capability with driver ability to extend the range of real situation utilisation of available performance.[11]

Further problems are incurred on 4 wheel drive systems sporting the ABS system. When braking on loose surfaces the wheel quickly stops rotating because it is effectively lubricated by the loose road surface. The ABS sensor immediately releases the wheel because it has stopped but before any significant braking can take place. Consequently the vehicle progresses along to the distress of the occupants without any significant slowing. The only current solution to this situation seems to be switching the ABS off completely.[12] Clearly this system would benefit from knowledge of the actual deceleration of the vehicle and the driver's requested deceleration from the brake pressure. From this extra information an intelligent decision could be made regarding the effectiveness of the ABS system.

It has also been found that even drivers trained on ABS in advanced driving schools do not always remember to apply the brakes hard enough to activate the ABS when they detect an incipient loss of control.[9]

Clearly, there are still many issues associated with the use of full time ABS. For the larger part, advantages of ABS tend to outweigh the drawbacks but in some areas such as those mentioned there is still a great deal of work to be done in creating systems that will perform at least as well as the current system, preferably better and never worse. At the heart of this intelligent decision-making is the real-time data upon which the decisions are based. There is a need for timely information about what the vehicle is doing and likely to do at every state point of a journey. Timely reliable estimation of the parameters responsible for roll over is the purpose of this thesis.

Automotive companies are looking for a solution that integrates these modern techniques as comprehensive tools for safer driving. These estimations provided in this thesis will form a solid basis for future work on intelligent automotive control systems that begin to address the problems outlined.

2.2 Control and prediction using applied intelligence

The first step to developing a control system is to be able to qualify and quantify the system in question. In the case of automotive stability control understanding and quantification of real time velocities and accelerations is paramount to all future work. In critical situations, built-in parameters allow a better vehicle behaviour estimation and the elimination of incorrect fault detection. In the case of complex systems, fuzzy logic and neural networks are often efficient tools to provide good results with reduced designing time.[13]

Porcel et al.[13] aimed to show that a neuro-fuzzy approach can be utilised to maintain response to critical situations with increased accuracy compared with classic methods. The test car was a front wheel drive (FWD) vehicle. It was fitted with sensors to measure longitudinal and transverse velocities by an optical cross-correlation sensor located at the rear of the vehicle. It also measured wheel velocities taken from the ABS system and gyro and acceleration sensors mounted near the centre of gravity of the car measured longitudinal and lateral accelerations and the yaw rate. The result of their investigation is outlined in the following sections.

2.2.1 Longitudinal Velocity Estimation

The critical driving conditions considered by Porcel et al. are as follows:

- Foot off the accelerator while cornering, as may occur when a driver drives too fast at the beginning of a bend, and reacts by taking the foot fully off the pedal while still cornering.
- Violent braking while cornering. In an emergency situation while cornering, the inner front wheel may lock, and the rear one may no longer adhere to the road. Both inner wheels may lose their contact with the road if the driver takes a bend on two wheels.

- Oversteering. This occurs when the vehicle becomes unstable during rear wheel lateral slide. The driver tries to compensate for the deviation of the vehicle from the normally expected trajectory by steering into the skid.
- Understeering. Front wheel slide out generally occurs when taking a sharp bend at a high speed and lateral sliding when a driver drives too fast into a wide curve near the grip limit.[13]

In the literature fuzzy logic was used to build two indicators to identify and detect the different ways a vehicle behaves these indicators show:

- **Loss of contact while cornering from lateral acceleration and yaw rate.** Lateral acceleration can be high without any rear wheel locking. This occurs at high yaw rate, ie. a tight curve. A number of rules determine the indicator value, if non-zero this detects the possibility of a rear wheel losing contact with the road.
- **Oversteering, front wheel sliding out and lateral sliding**. Detects critical deviations from the intended driving behaviour of the vehicle during lateral motion. Steering wheel angle, lateral acceleration, the derivative of yaw rate, and the derivative of transverse acceleration are considered to do this.[13]

These indicators are discussed in more detail in chapter 6. The aggregation of the system means that the best combination of wheel speeds or the integral of the longitudinal acceleration can be used at any one time to determine the longitudinal velocity.

2.2.2 Longitudinal Velocity Results

The results for the longitudinal velocity show an improvement of the proposed fuzzy estimator over traditional methods of determining longitudinal velocities. Figure 2.2-1 shows a very slight improvement in accuracy with the foot off the accelerator while cornering.

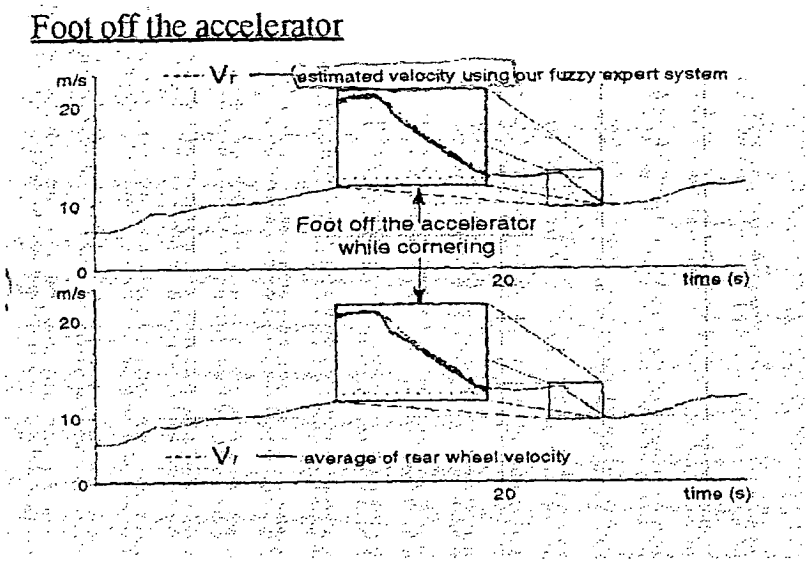


Figure 2.2-1 Comparison of results for a test cycle, on a wet road, in "foot off the accelerator position", while cornering to the right.[13]

The improvement is more pronounced in a small section of the 'violent braking while cornering' as depicted in Figure 2.2-2. Both comparisons are with the average of rear wheel velocities and the graphs show the difference in errors (up to 8 m/s in the violent braking case). This is based on the inside wheel locking up during braking and cornering which means one of the wheel velocities will be zero. Thus the average will be significantly out even if one of the wheel speeds is correct.

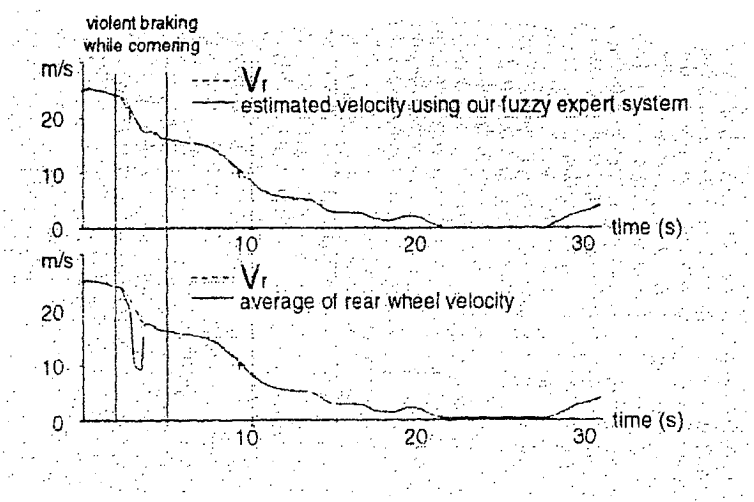


Figure 2.2-2 Comparison of results for a violent braking cycle while cornering.[13]

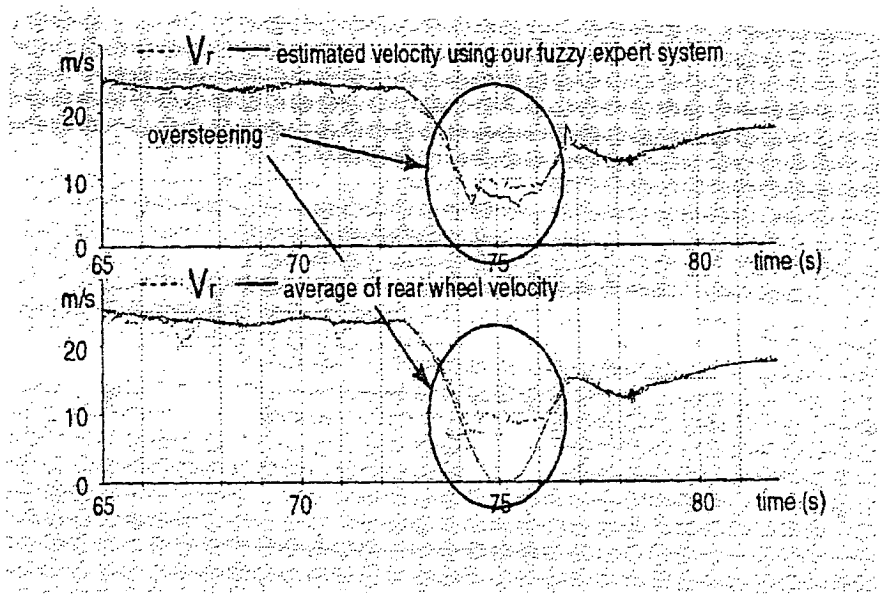


Figure 2.2-3 Comparison of results with the fuzzy estimator using the wheel speed only for oversteering.[13]

Considering Figure 2.2-3, we see the results of an oversteering case where the front wheel is sliding out and lateral sliding is induced. Once again the difference between the estimation based on average wheel speeds is out by up to 8 m/s. The estimator on the other hand is much more accurate giving a result to within 1.5 m/s. Clearly any system that is based purely on wheel speeds may become grossly inaccurate in the critical conditions when it is most needed to prevent an accident. The results presented thus far were used to demonstrate the effectiveness of a fuzzy logic system in this type of application. These results are used to estimate longitudinal velocity in Chapter 6 as a parameter considered responsible for vehicle roll over. The following section outlines the work done by Porcel et al. on transverse velocity estimations using neural networks. It is included to support the use of neural networks as a tool in the estimation of vehicle dynamics.

2.2.3 Transverse Velocity Estimation

There are no means to measure the transverse velocity in a direct way with the help of low cost sensors. Therefore it was necessary for Porcel et al.[13] to create an estimation system based on inputs from the ABS, the gyroscope sensor, the acceleration sensors and the steering wheel angle. With a fuzzy inference system the

longitudinal velocity was reconstructed. This system served to detect wheels' losses of contact, front wheel sliding out and the vehicle's lateral motion. This information resulted in the choice of different wheel speeds to be referred to as the longitudinal velocity.

While the vehicle response for this investigation was known, the unknown friction coefficients between tyre and road surface and tyre characteristics made the case complex. The use of neural networks simplified the task immensely, as the unknown details no longer require modelling or approximation. By feeding the neural networks with sensor outputs, the estimated longitudinal velocity and the transverse velocity as measured by the cross-correlation sensor, the network was trained. Porcel et al.[13] decided to break the cases down into a number of instances typical of the vehicle's behaviour under critical conditions and use these to each to train a network. Thus a number of networks were developed and a fuzzy controller used to determine which of the networks was most appropriate given the vehicle's dynamic state.

The three fields concerning transverse velocity were identified as:

- Understeering, even high system inputs (transverse acceleration, steering angle) lead to a relatively small system output (transverse velocity)
- Oversteering, rear wheel lateral slide or even total vehicle instability cause high transverse velocities.
- Braking in a turn, a locked inner front wheel and a sliding inner rear wheel may cause vehicle instability. The resulting yaw rate is responsible for high transverse velocities.

2.2.3.1 Neural Network Representation

Porcel et al.[13] tested a number of different networks examining combinations of input variables and testing performance for precision and case representation. The combinations of input variables compared included longitudinal and transverse acceleration, the steering wheel angle, longitudinal velocity, yaw rate, the derivative of yaw rate and the output in the last time step.

The networks considered were feed-forward networks, radial basis networks and Elman networks. In this investigation the Elman nets were found to display characteristic saturation effects, when confronted with inputs that were higher than those appearing in the training data. The performance of the radial basis nets was similar to that of the feed-froward nets, but required more neurons. (Please refer to chapter 3 for a comprehensive discussion of neural network theory and practice.)

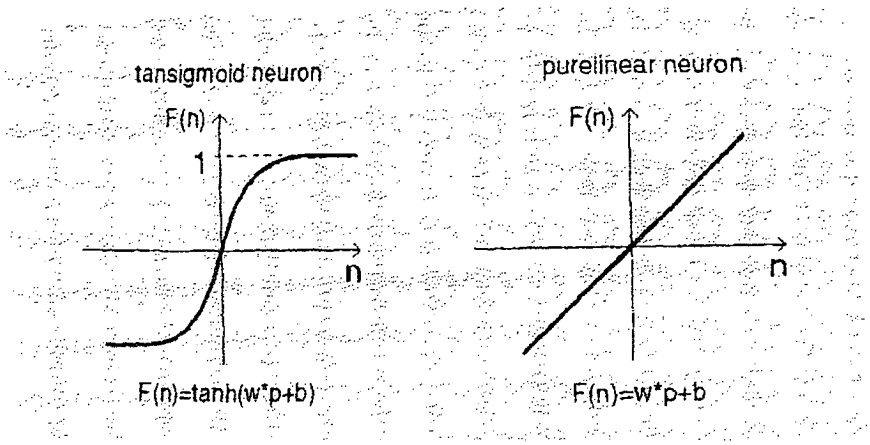


Figure 2.2-4 Neuron transfer functions: tansigmoid and purelinear.[13]

Activation functions considered were logsigmoid, tansigmoid and radial basis neurons, with the final choice lying with tansigmoid in the input and hidden layer and one purely linear neuron in the output layer (see Figure 2.2-4). The networks were trained and tested using standard procedures (outlined in chapter 3).

The results of the network training and testing were presented as follows for each of the three cases.

2.2.3.2 Understeering conditions

The results for the test involving understeering conditions are shown in Figure 2.2-5. The results are based on longitudinal acceleration, the derivative of transverse acceleration, yaw rate and its derivative; and longitudinal velocity. Clearly, these are only a few of many possible variables that could have been used. A blanket approach, using all variables that could possibly contribute to the transverse velocity would be the next step toward improving the results.

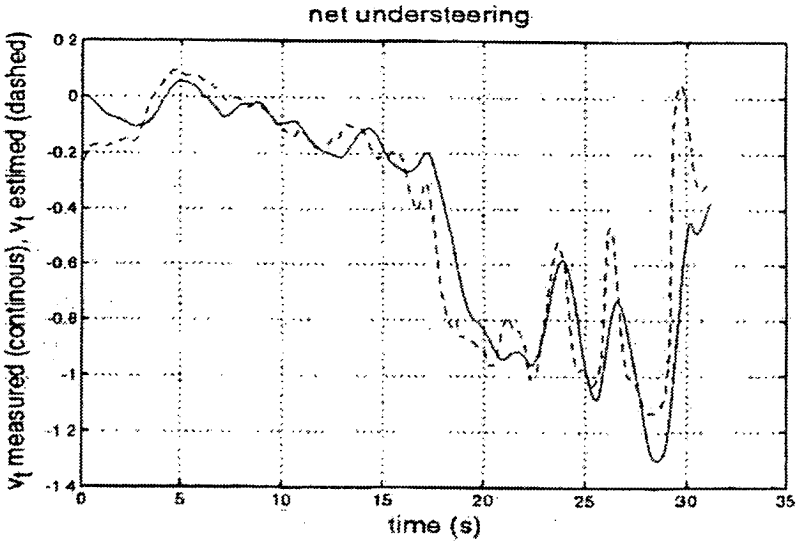


Figure 2.2-5 Transverse Velocity: real and estimated value for understeering.[13]

2.2.3.3 Oversteering conditions

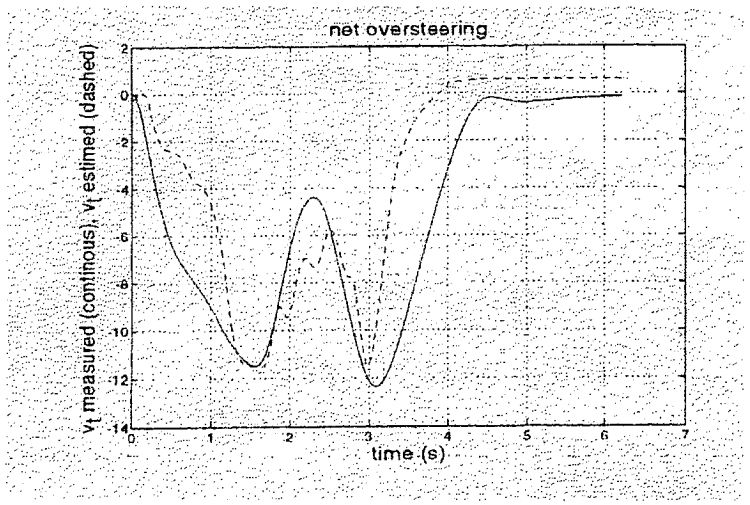


Figure 2.2-6 Transverse Velocity: real and estimated value for oversteering.[13]

We can see from Figure 2.2-6 that the network gives an overview of the general trend associated with the transverse velocity variation. There is a tendency to smooth out or generalise the actual results. This is a trademark of the neural network approach. It may be possible to improve the results by further training the network but it is likely that further training would simply lead to over fitting, whereby the errors and anomalies in the measurements are treated as part of the trend and the curve over complicated. The inputs used to train the network were the same as the previous section.

2.2.3.4 Braking in a turn

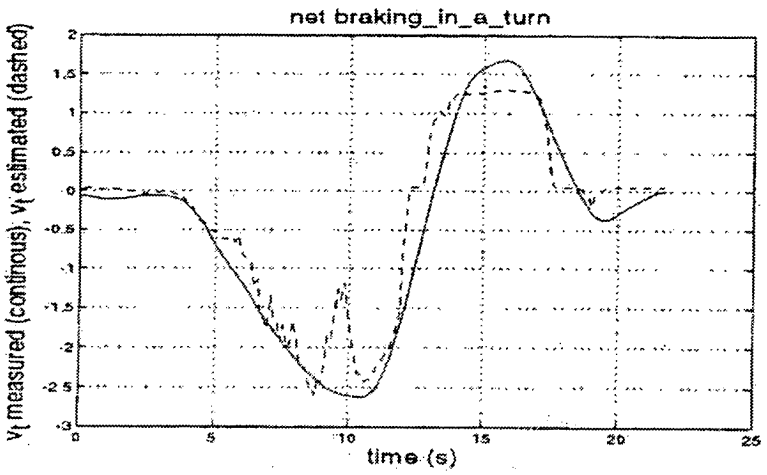


Figure 2.2-7 Transverse Velocity: real and estimated value for "Braking in a turn".[13]

The results for "braking in a turn" as shown in Figure 2.2-7 are much more encouraging. Once again some of the detail is missing but the overall trend is in agreement. Again, future work would aim to improve the results through the use of different neural networks or an increased number of inputs.

2.3 Applied Intelligence to Automotives

The following is a broad view of applied intelligence to automotives, while not directly related to the task at hand, it is included for academic interest, with specific details of this particular study to follow.

2.3.1 Engine related

Applied intelligence has found its way into the automotive industries particularly in those areas relating to the engine. This is combined in some instances with fuzzy logic technology to produce some interesting work. Applied intelligence in this area falls into two main areas, engine control and engine diagnostics.

2.3.1.1 Engine diagnostics

Lu, Chen and Hamilton outlined the development of a fuzzy diagnostic model for automotive fault diagnosis.[14] The model uses a fuzzy rule generation algorithm

developed by the team and is based on priority rules. The model has been implemented in a vacuum leak detection agent system and after testing on large sets of data was proven to be excellent. The system is being implemented into a Ford test system for end-of-line test.

Virtual sensing is another promising technique in this field. A neural network based engine performance, fuel efficiency and emissions prediction system has been developed for Spark Ignition (SI) and Compression Ignition engines (CI).[15] Through limited training on an engine dynamometer the network is able to accurately predict real-time engine power output, fuel consumption and regulated exhaust emissions over highly transient engine operating conditions and using only readily measured engine parameters. The instantaneous prediction of exhaust emissions may form the basis of an intelligent diagnostic system although it appears this has yet to be attempted by this group.

2.3.1.2 Engine Mapping

The characteristics of neural networks are such that they are ideally suited to control situations particularly where the input may be noisy. This is particularly true of the work done in the field of engine control applications. Neural networks provide a simple and effective way to describe the linear input-output behaviour of a system through 3-d mappings. The limitations of the system have been examined and were found to be such that an advantage over conventional look-up tables varied from case to case.[16] The main contributing factors to the decision were the amount of memory available in the control module, allowable computation times, need for the on-line adaptation capability, and required transparency of the representation. The neural network model has much smaller memory requirements at 31 parameters than a full interpolating look-up table. Where adaptation was required the neural network model was also superior.

2.3.1.3 Application to Idle Speed Regulation

Neural network based, discrete adaptive sliding mode control has been developed for idle speed regulation in internal combustion engines.[17] The control goal in this situation was to lower the idle speed for better fuel economy while rejecting typical

load disturbances and reducing engine speed variations for drive comfort. This approach was simulated using a Ford V8 4.6L engine. Feed-forward neural networks were used to approximate the unknown system dynamics. In this case the Network provided a number of advantages over conventional approaches. These were:

1. explicit knowledge of the system dynamics is not necessary for the controller design;
2. the controller is adaptive to the system parameter uncertainties and external disturbances;
3. the time delay in the system can be addressed by increasing the relative order of the system.

They succeeded in showing that it is possible to use neural network based technology for the efficient control of idle speed and that it can be a simpler tool to use than conventional methods.

2.3.1.4 Air-fuel ratio control for directly injected spark ignition engines

Spark ignition engines that are directly fuel injected tend to run lean at low loads and stoichiometric for the other operating areas. Lenz and Schroder have shown that a normalised Radial Basis Function artificial neural network can be used as a mapping from the driver's input and operating conditions of the engine on injection commands to achieve a defined air-fuel ratio in transient operation.[18] This network is trainable and represents an intelligent feed-forward control structure. It therefore takes account of wear and alters the mapping accordingly. Hence the best performance is maintained at all times.

In addition to what has been done at a practical level, artificial neural networks have also been used to model the behaviour of the spark ignition engine from the perspective of an input/output system. An example of this is given by Lichtenthaler et al.[19]. Hardware-in-the-Loop (HIL) simulation was used to support test and verification during the development phase. The neural networks were shown to have advantages with respect to robustness and measuring extent. They could also be used as stand alone models or as sub-models integrated in a global model based on a physical structure.

Various methods have been applied to the question of vehicle dynamics. Largely the need for modelling arises from Computer aided engineering or CAE. With an accurate estimate of the various stresses and loads that will be present in vehicle dynamics, the vehicle may be engineered to be as light and as strong as both is possible and necessary.

This concept is by no means new and was under development in 1989.[20] Previous approaches have been complex and computationally intensive. They have also required a great deal of knowledge and experience to be of real use. Two techniques that have warranted research in this area of automotives are bond graph modelling and the use of Kalman filters.

2.3.1.5 Vehicle control

In the area of vehicle control neural networks and fuzzy logic have both played a part. A good example of the possibilities that also serves to demonstrate the strengths and weaknesses of each approach is the fuzzy and neural truck backer upper control systems.[21] The comparative study was tested in a computer simulation environment not on real world data. A fuzzy logic rule base was set up from empirical know-how to reverse the truck model into a docking bay. The comparison was made with the Truck backer upper proposed by Nguyen and Widrow.[22] Some of the results are shown below:

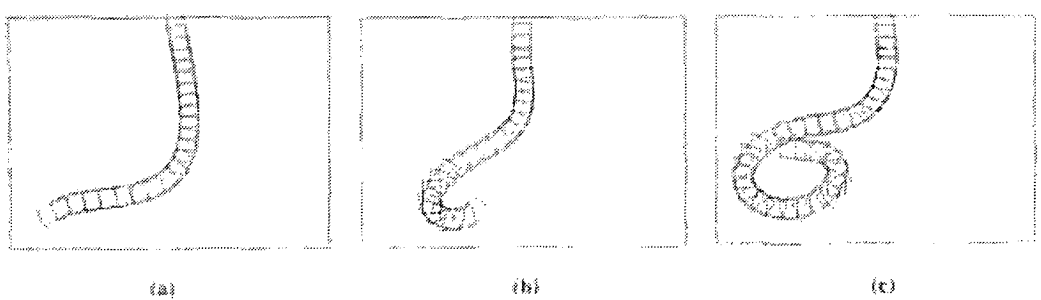


Figure 2.3-1 Sample truck trajectories of the neural controller for initial positions (x, y, ϕ): (a) (20, 20, 30), (b) (30, 10,220), and (c) (30, 40, -10).[21]

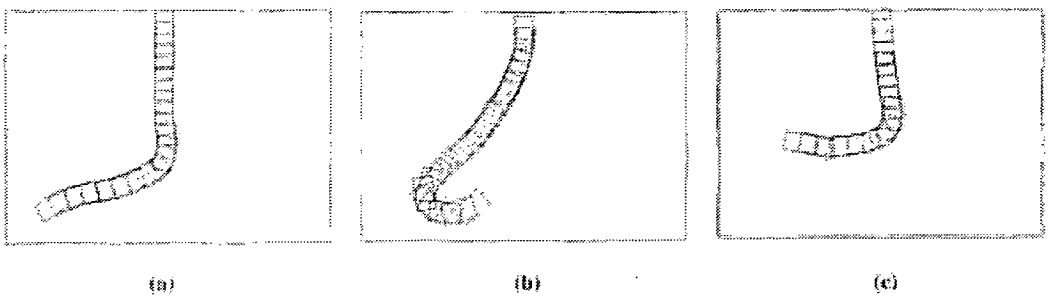


Figure 2.3-2 Sample truck trajectories of the fuzzy controller for initial positions (x, y, ϕ): (a) (20, 20, 30), (b) (30, 10,220), and (c) (30, 40, -10).[21]

Figure 2.3-1 traces the path of the truck as the neural network controls the backing path. In Figure 2.3-1 (a) and (b) the path is very close to the fuzzy logic result. However comparison with (c) in Figure 2.3-2 shows that the neural network result is far from optimum. In fact the simulated truck does a complete 360 degree turn whilst backing into the docking bay.

The fuzzy approach here was found to have advantages over the neural network approach - based on a back-propagation algorithm. This is not surprising, as fuzzy rules are able to replicate common sense type rules. The real advantage of the neural network approach becomes more evident when the complexity of the system is outside of the scope of common sense and a simple rule base. In this respect the neural network would be more powerful in a more complex situation where the empirical know-how is not as easy to come by. Similarly, the neural network approach is ideally suited for applications where smooth continuous results are required, Figures 2.3-1 and 2.3-2 begin to show the smoother operation of the neural network result.

The fuzzy rule based model was also found to be more robust as rules were removed than the neural network was when training data was reduced. This would suggest that the neural network algorithm was less than ideal. It is interesting to note however that once trained a neural network no longer requires its training data where as the fuzzy rule system will always require each rule. Hence the neural system is less likely to lose the network it depends on than the fuzzy rule system. The results between the two were extremely close up to 50% loss of information.

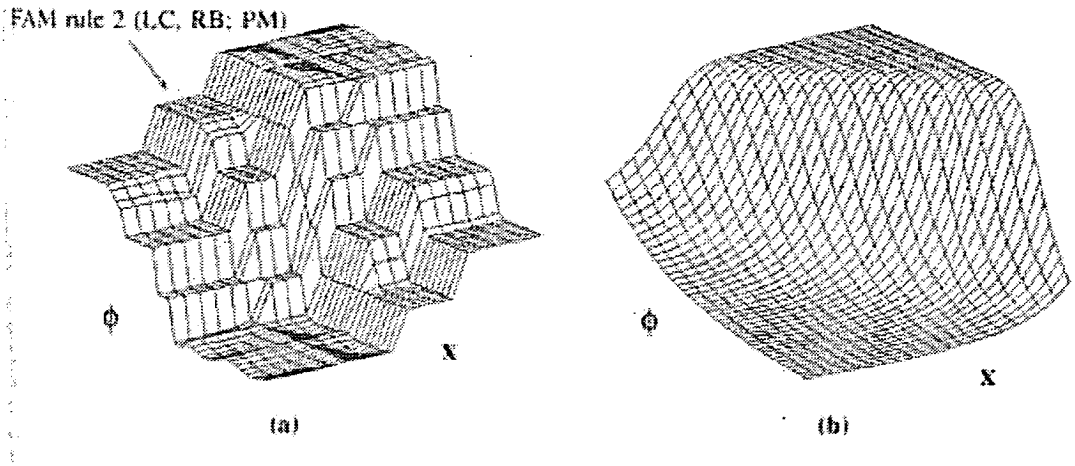


Figure 2.3-3 (a) Control surface of the fuzzy controller, Fuzzy set values determine the input and output combination. (b) Corresponding control surface of the neural controller for a constant value y = 20.[21]

The fundamental difference between the two methods is detailed in Figure 2.3-3. The fuzzy rules are relatively discrete, meaning that there will be large jumps between consecutive points. The neural network control system, on the other hand, is perfectly smooth. This allows the network to interpolate and give very smooth variation of the output with the input variation. In very sensitive applications such as complex dynamics control and estimation, this quality is essential.

2.4 Other Neural Network Applications

While the field of neural networks is continuously growing it would seem that the actual volume of practical research is quite small compared to the number of simulations and models based and tested on entirely theoretical values. While it would appear that the theory is indeed well developed and the applications apparently viable, there is conspicuously small number of practical, real life working applications. The clear majority of work to date in the automotive area has focussed on modelling a particular performance rather than actual application to a real time process. This is generally done with computer simulation, and whilst proving the result to be possible, it falls short of actual practical application.

Some of the areas where fuzzy logic and neural network tools are finding applications include:

1. quality prediction in industrial control [23, 24],
2. manufacturing applications, such as design, process planning and scheduling [25, 26],
3. process control, fault diagnosis and condition monitoring [27, 28, 29, 30],
4. control and monitoring of machining processes [31, 32, 33, 34],
5. pattern recognition [35, 36, 37], and
6. robotic control [38, 39]

One area of research, that has proven to be ideally suited to artificial intelligence applications, is manufacturing. The complex non-linear nature of manufacturing and process control means that it is ideally suited to neural network modelling. A general indication of this field was included to demonstrate the success possible through use of neural networks.

2.5 Concluding Remarks

If we consider the state of the art control and safety systems on new a concept vehicles such as: Anti-lock braking (ABS) [8], Vehicle Dynamic Control (VDC) [7] and Dynamic Stability Control (DSC) from BMW [40], Traction Control (ASR) [41], All Wheel Drive (AWD), Tyre Pressure monitoring system [42], Dynamic Brake Control (DBC)[43] and Brake by wire [44], it is clear that these developments are focussed on localised control. DSC and VDC are moving toward a comprehensive system but there is no one vehicle that incorporates all of the design features available to date. Hence the need still exists for a comprehensive intelligent control system based on a complete, real time knowledge of the individual vehicle's dynamic behaviour.

The state of dynamic performance prediction is well developed in a theoretical sense. Models have been developed that demonstrate the effectiveness of a particular system in simulation mode. There are very few systems that can produce dynamic performance prediction. A need exists for the estimation of salient data to form a basis for the comprehensive control outlined above.

Velocity as a parameter responsible for overturning is currently estimated using first principles. This does not take into account the non-linear forces associated with the deformation of tyre walls or the unusual friction conditions associated with skidding. For control purposes there is a need to estimate the parameters responsible for vehicle overturning under all road conditions.

Reliable estimation of acceleration in 3 dimensions would lead to more accurate modelling for stress analysis and safer more efficient design of safety systems and other vehicle systems. Estimation of these values lays the basis for accurate, comprehensive and intelligent control systems that will eventually lead to greater safety and efficiency in the automotive industry.

In conclusion, the state of the art for reliable quantitative estimation of parameters responsible for vehicle overturning is inadequate. This work will propose a possible solution to rectify this inadequacy through the use of established neural networks and modification for estimation of these performance parameters. The following chapters will outline the networks to be used, the experimental set up and results.

Chapter 3 Computational Intelligence Literature Review

In considering the application of artificial neural networks to prediction of parameters responsible for vehicle roll over, it is important to have a good basic understanding of the background theory and some common terminology. This chapter is included to give a brief overview of Artificial Intelligence and artificial neural network theory as a basis for the work detailed in chapter 6.

3.1 What is Artificial intelligence?

Artificial intelligence and computational intelligence are terms used to cover an ever-increasing field of research. Artificial Intelligence or AI has come to mean many different things. Martin Fischler and Oscar Firschien [45] describe three theories developed in philosophy. These theories are the "existence theories"

1. Intelligence is a non-physical property of living organisms and cannot be recreated in a machine
2. Intelligence is an emergent property of organic matter: silicon is inadequate, but when we learn how to build machines out of organic compounds there may be a chance of developing intelligent behaviour.
3. Intelligence is a functional property of formal systems, and is completely independent of any physical embodiment.

Engineering is primarily concerned with the third viewpoint. Intelligence can then be defined on a number of attributes including the ability to learn, recognise symbols, speak and understand speech. For the many different attributes, there exist computer-based simulations that display their characteristics to some degree. Although there has yet to be developed a system that will encompass all of the ideals put forward here, a number of different disciplines have been developed. These include expert systems, natural languages, and simulation of human sensory capabilities (eg image recognition), robotics and artificial neural networks. As the main area of interest is predicting complex non-linear relations a suitable tool for the task at hand is the artificial neural network. This will be the main focus of this outline.

3.2 The Biological Neuron

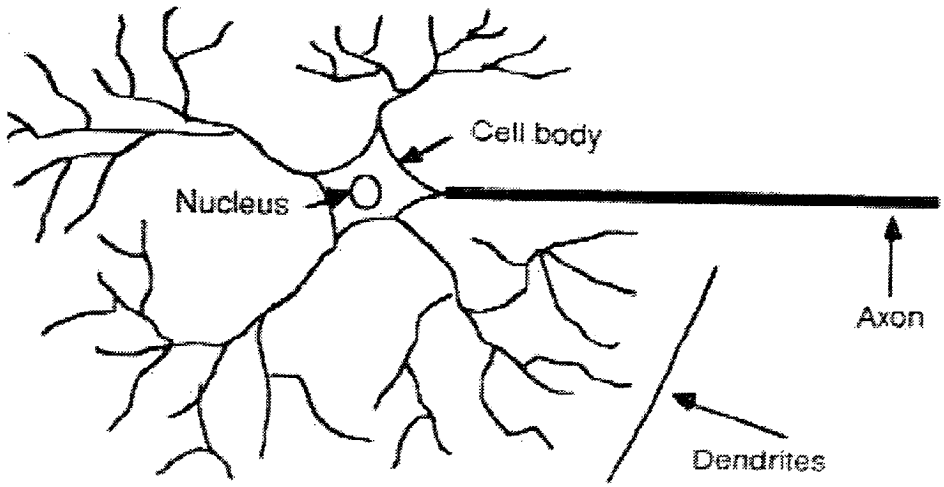


Figure 3.2-1 Biological Neuron [46]

A neuron is the biological building block of the brain and the biological neural system as represented in Figure 3.2-1. It allows inputs from a large number of similar neurons. It will then send a single output to many other neurons across the network of axons (neuron connectors). Each axons has many branches known as axon collaterals that join each end at the input of another neuron in a connection known as a synapse shown in Figure 3.2-2. The parts of the neuron to which these synapses are joined are called dendrites. Many synapses are fixed but many are adaptive or plastic which means that they can increase or decrease in strength under appropriate conditions. As such these synapses have differing strengths or synaptic weights. They come in two forms which allow either of two effects on their associated neuron, excitatory (positive) and inhibitive (negative). Finally synapses are unidirectional, in that they allow signals to pass in only one direction.[47]

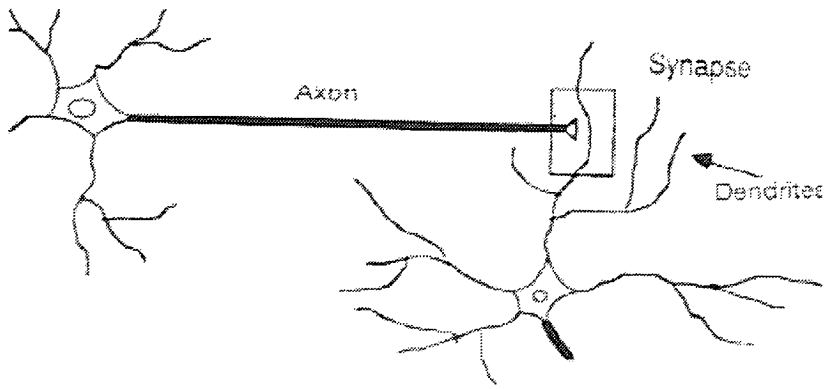


Figure 3.2-2 Biological Synapse [46]

The physical assumptions that many artificial neural networks are based upon are given in Figure 3.3-1

"1. The activity of the neuron is an "all-or-none" process.
2. A certain fixed number of synapses must be excited within the period of latent addition in order to excite a neuron at any time, and this number is independent of previous activity and position on the neuron.
3. The only significant delay within the nervous system is synaptic delay

Figure 3.3-1 Artificial Neural Network Assumptions

In the McCulloch-Pitts model the artificial neuron provides the weighted sum of its input potential's in a numeric value. The neuron then will fire or not dependent on whether the summed weighted input is greater than a threshold value. If it does fire it will transmit this summed weighted input value, if not it will transmit nothing.

Schematically, this set-up may be represented as in Figure 3.3-2.

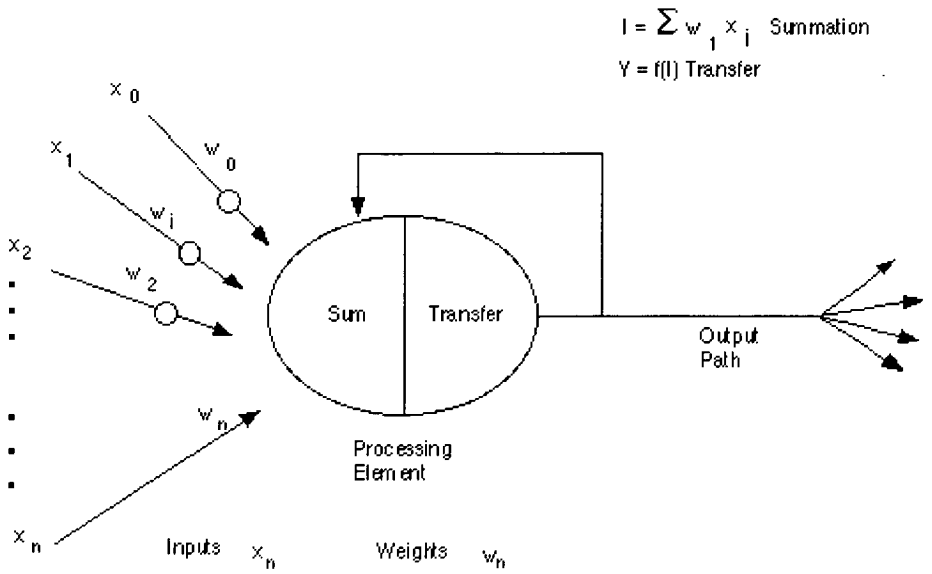


Figure 3.3-2 Basic Structure of an Artificial Neuron [64]

3.4 Neurons as functions

At the most basic level there are three functions that combine to give the neuron its processing capability. These functions are the input function, activation function and output function. All three functions may be combined to into one function known as

3.3 A Brief History of Neural Networks

The first modelling of neurons dates back to the 1940s when McCulloch and Pitts published their classic paper "A logical Calculus of the Ideas Immanent in Nervous activity".[48] The original neurons were simple logic gates AND, OR and NOT. Their work forms much of the current basis for neural network development. In 1949, Donald Hebb introduced the Hebbian learning rule, a learning scheme that purported to store information in the connections.[49]

Frank Rosenblatt invented the Perceptron in 1958 [50]. As will be described later, this model was capable of learning patterns by modifying connections to the threshold elements. In the early 1960's Bernard Widrow and Marcien Hoff introduced ADALINE (ADAptive LINEar combiner) [51]. This was a basic pattern recognition device from which the Widrow-Hoff learning rule developed. (In this rule, the summed square error is minimised during training.) In 1965 Nils Nilsson [52] summarised the developments of the time, formulating inherent limitations of learning machines with modifiable connections. At this time layered networks existed but no efficient learning techniques had as yet been developed.

The period from 1965 to 1984 proved quiet and there was little research done to improve upon the work done to date. Learning of threshold elements was studied by Sum-Ichi Amari [53, 54]. The neural architecture for visual recognition known as neocognition was developed by Kunihiko Fukushima [55]. Tuevo Kohonen [56,57]; pursued associative memory research during this period, while Adaptive Resonance was worked on by James A. Anderson [58]. A number of neural architectures were introduced by Stephen Grossberg [59,60]. Recurrent neural architectures for associative memories were introduced by John Hopfield [61,62] and rekindled much of the interest in neural networks. The publication of "Parallel distributed processing" by James McClelland and David Rumelhart [63] brought the field back into main stream focus in 1986. The theories and learning rules introduced began to show the true potential of the once dubious layered networks.

the transfer function although it is useful to consider them separately for the purposes of simplicity.

3.4.1 Input function

Fundamentally, the input function provides a summation of the multiplication of inputs with the corresponding weights. It generates the input for a neuron from the outputs from every other neuron connected synaptically to it. The function multiplies them by their corresponding weights and then sums the result to create the input for the activation function.

This may be written as

$$net_j = \sum_i x_i w_{ji} \qquad \textbf{Equation 3.4-1}$$

where i denotes the number of input neurons.

3.4.2 Activation function

The activation function is non-linear. The purpose of this function is to determine the output of the neuron. The activation function may be any function that is both monotonically increasing and differentiable. The output range of the activation function is usually limited to between 0 and 1 or −1 and +1. Early models used a simple threshold function or step function for this purpose as shown in Figure 3.4-1. The result being that a neuron would output a value of 1 if the inputs exceeded the threshold value or zero if it did not.

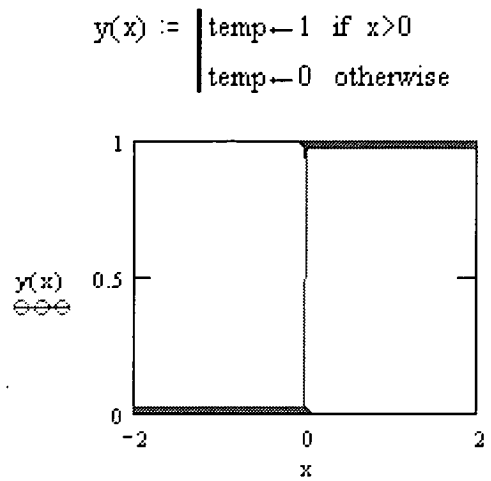


Figure 3.4-1 Graph of Threshold or Step function [65]

However, a more general, non-linear function known as the sigmoid function has been used more recently for this purpose. The function may be loosely defined as a continuous, real-valued function whose derivative is always positive, and whose range is bounded. This function may be represented by Equation 3.4-2 and graphically as shown in Figure 3.4-2.

S(x) = 1/(1+Exp(-cx)) **Equation 3.4-2[66]**

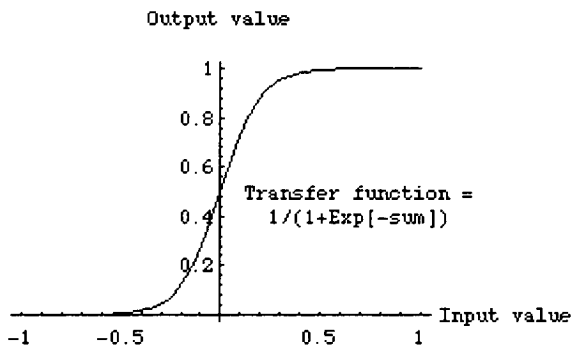


Figure 3.4-2 Sigmoidal Activation Function [65]

A major advantage of the sigmoidal function is that its derivative is relatively easy to calculate, important in the neural scheme of things. It is given by:

S'(x) = S(x) *(1-S(x)) **Equation 3.4-3[66]**

3.4.3 Output function

The final component of the transfer function is generally chosen to be equal to the output of the activation function or, in other words, the output of the neuron will be the same as the activation. So, the sigmoid function represents the neuron transfer function with the horizontal axis representing the weighted summed input and the vertical axis representing the neuron output.

3.5 Perceptron

The Perceptron is a pattern classification system developed by Frank Rosenblatt in the late 1950s, early 1960s.[50] Rosenblatt became fascinated with the operation of the eye of a fly. Much of the processing done to tell the fly to escape is done in the eye. The single layer Perceptron was found to be useful in classifying a continuous-valued set of inputs into one of two classes. As such the Perceptron recognises abstract and

geometric patterns from optical input patterns disregarding noise in the input. The neuron computes the weighted sum of the input signals and compares that net weighted input to a threshold value, T. If the net input is greater than or equal to the threshold, the neuron outputs +1, if not, it outputs –1. This initial model was further developed by Minsky et al.[67] a generalised description follows.

3.5.1 Simple two-layer Perceptron.

Figure 3.5-1 shows the architecture of a simple two layer Perceptron. There is one layer of input nodes and one layer of output nodes. Each of these two layers is fully connected to the nodes in the other layer but to none of the nodes in its own layer. When a signal is sent from the input layer to the output layer, it has the corresponding weight applied to it and the receiving node in the output layer sums all the values it receives. If this sum exceeds a given threshold, that node in turn will produce an output signal, otherwise it remains dormant.

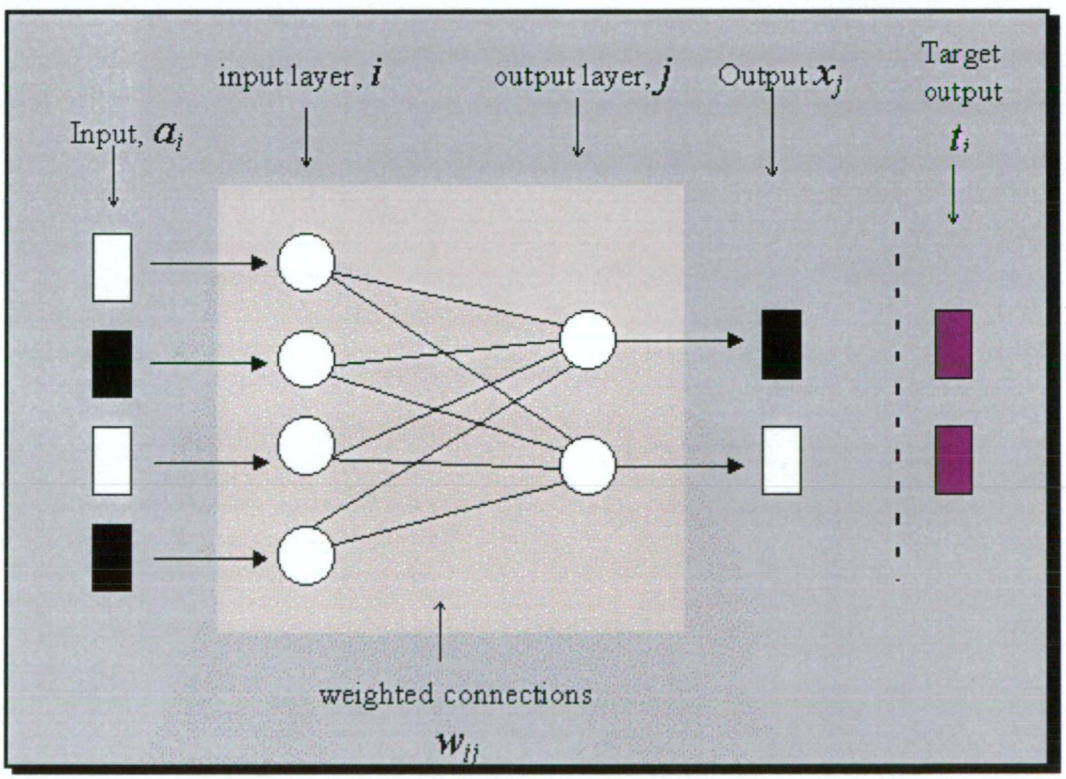


Figure 3.5-1 Simple Perception layout [68]

The output of a node may be expressed as:

$$S_j = \sum_{i=0}^{n} a_i w_{ij}$$

If $S_j > \theta$ then $x_j = 1$ **Equation 3.5-1**

If $S_j \leq \theta$ then $x_j = 0$

Where θ is a predetermined threshold value

The Perceptron may be trained to produce the desired output by adjusting the weights through use of the following training algorithm.[69]

$$w_{ij}^{new} = w_{ij}^{old} + C(t_j - x_j)a_i$$

Where C is a learning rate constant

x_j is the actual output **Equation 3.5-2**

t_j is the desired output

a_i is the input node output

The functional limitation of the Perceptron is that it can only recognise linearly separable patterns owing to having only one adaptive layer. (A linearly separable pattern is one that can be separated into two distinct classes by drawing a single line.) In demonstrating that this can be done it is something of an historical landmark in the history of neural networks.[67]

3.6 Types of Networks

A great many neural networks have been developed over the years, each is suited more or less to a particular type of problem. Figure 3.6-1 shows a taxonomy or family tree of some of the better known networks. Each of the networks can be training to solve a specific problem using compilations of training data. Artificial neural networks are often described as a black box; this is largely because the process is self-organising. The desired output is provided but the way in which this outcome is achieved is left to the internal algorithms. In general, a neural network will map a real input of any given dimension to a real output of some other given dimension. It is worth doing a brief tour of some of the types and the purposes to which they are best suited. The major difference is between the feed-forward and recurrent model types. Beyond this, models are delineated by the training method as supervised or unsupervised.

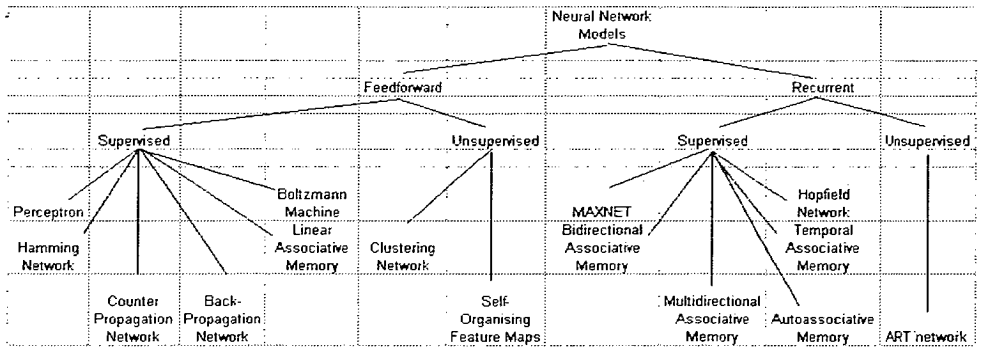


Figure 3.6-1 A Taxonomy of Neural Network Models [70]

3.6.1 Feed-forward Networks

The Perceptron, as has been shown, is a prime example of a feed-forward network. Primarily these networks are distinguished by the passage of input signals in one direction only. The signal makes its way from the input layer through the weighting system to the output layer where it is compared to the required result and the weights modified. The process then repeats. In its general form the feed-forward network has one input layer to receive information from the knowledge base, any number of hidden layers (the processing power) and one output layer that passes information to the surrounding environment. Each layer in the network contains neurons that receive any number of inputs and send a single output to other neurons in the following layer.

3.6.2 Recurrent Networks

The recurrent network differs from the feed-forward net in that it has at least one feedback loop. The classic example of this type of network is the Hopfield model, first introduced by John Hopfield [61]. The recurrent network may consist of a single layer of neurons with each neuron feeding its output signal back to the inputs of all the other neurons. The feedback implied by this system creates a dynamic response, allowing the network to evolve as time elapses. If a network is stable, eventually, it will reach equilibrium; however, instability can be a troublesome aspect of this type of network. A conceptual comparison of the two network types is shown in Figure 3.6-2.

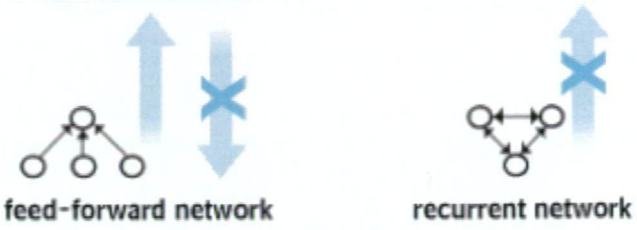


Figure 3.6-2 Conceptual comparison of feed-forward and recurrent networks[71]

3.6.3 Some Important Characteristics of Neural Networks

Neural networks are very different from other mapping techniques in a number of aspects. These differences are what make them so unique as a tool, a list of the important characteristics follow:

- Neural networks consist of a numerous, very simple processing elements (neurons) that communicate through a rich set of interconnections with variable weights or strengths.
- Information (memories) are stored or represented by the interconnections between the neurons. Information is processed by a spreading, constantly changing pattern of activity across many neurons.
- Neural networks are trained rather than programmed, there is no imperative to completely understand a system to be able to replicate it. Some systems are capable of autonomous learning and some of learning by trial and error.
- Neural networks do not consist of separate memory and controller, plus externally stored program to dictate the operation of the system as in a digital computer. The neural network is controlled instead by 1. The transfer function of the neurons, 2. The detailed structure of the connections amongst the neurons, 3. The learning law the system follows.
- Neural networks act as associative memories; they group similar items together within their structure. As a memory, a neural network can retrieve stored information from incomplete, noisy, or partially incorrect input cues.
- Neural networks are able to generalise, that is learn the characteristics of a general categories based on a series of specific examples from that category.

- Neural networks are highly fault tolerate and are capable of continuing to function after a significant proportion of its neurons and interconnections have become defective. Its performance will degrade slowly and smoothly as neurons and interconnections fail.
- Neural networks innately act as a processor for time-dependent spatial patterns or spatiotemproal patterns.
- Neural networks can be self-organising. Certain networks can be made to generalise from data patterns used in training without being provided with specific instructions on precisely what to learn.

The preceding list summarises the basic functions of neural networks and highlights some of the desirable features that are contributing to their growing application popularity.[24]

3.7 Considerations for Improving and Evaluating Network Performance

There are a number of considerations for improving and evaluating neural network performance. Among these is the type and preparation of data that is to be used as well as the choice of network architecture, which is covered in section 3.8. Testing of the network is used to evaluate its performance.

3.7.1 Design of Network Training Data

To maximise the usefulness of a particular network it is important to carefully consider the type of data with which it is to be trained, as well as how that data is to be prepared. Testing is also important to determine the effectiveness of the network. The performance of a network is dependent on the values used to train it. Most significant is the number of values used in the training set. The two goals a data set must satisfy are [72]:

1) Every variable in the training data set must be adequately represented. Usually, the training data will consist of several possible subgroups, each having its own central tendency toward a particular pattern. All of these patterns must be represented sufficiently.
2) Within each class, statistical variation must be adequately represented. It is the presence of random noise imposed onto pure patterns that makes most neural

network applications necessary. The training set must be designed to insure that an adequate variety of noise effects are included.

3.7.2 Normalising Network Input

Network performance can often be improved by removal of insignificant characteristics such as standard deviations and offsets which, can serve to obscure the real issue. Hence, scaling the network, otherwise known as normalising can serve to remove certain insignificant characteristics from the training set by placing all values within certain predetermined limits. This avoids the network creating a bias toward vector components that are of higher magnitude. Normalising also creates an equal footing for error minimisation, where greater magnitude for certain elements would dominate the correction process as it is based on the total error from all outputs. It is imperative to normalise the target output when the activation function used results in a bounded output. For example the sigmoid function outputs a value between 0.0 and 1.0, for this particular function the target output must also be between these values for the network to train appropriately.

Data is commonly normalised as follows [24]:

$$norm(x_i) = \frac{x_i - \min(x_j)}{\max(x_j) - \min(x_j)} \qquad \textbf{Equation 3.7-1}$$

where $norm(x_i)$ is the normalised i^{th} value in a set of j values,

x_i is the original i^{th} value in a set of j values,

$\min(x_j)$ is the original minimum value in a set of j values, and

$\max(x_j)$ is the original maximum value in a set of j values.

The data is thus scaled over 0.0 – 1.0 such that the original minimum value becomes equal to 0.0 and original maximum value becomes equal to 1.0, the values in between are uniformly scaled.

3.7.3 Network Testing and Performance

Initially, the network is trained on a set of values, so the first step is to test that it can accurately predict the values it has been trained on. Following this the network must be tested on its ability to interpolate or generalise, that is, to predict data from values that it has not yet specifically seen.[24]

3.8 Network Architectures for Decision Making

It has been found that Back Propagation (BP) and Radial Basis Function (RBF) networks have been used for applications similar to the predictions of parameters to prevent vehicle roll over, as discussed in the previous chapter. In this work both BP and RBF will be used for prediction purposes with minor modification. This section details the algorithms for both modes with the relevant code shown in the relevant section of Appendix A – Source Code.

3.8.1 Back Propagation Neural Network (BP)

The architecture of the BP network is similar to that of the Perceptron. It consists of an input layer, one or more hidden layers and an output layer. There are *i* input nodes, *j* hidden nodes and *k* output nodes. All input nodes are connected to all hidden nodes through weighted connection, w_{ij}, and all hidden nodes are connected to all output nodes through weighted connection, w_{kj} as shown in Figure 3.8-1

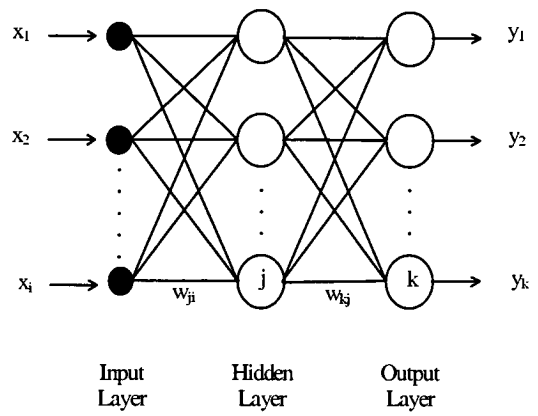


Figure 3.8-1 Back Propagation Neural Network Architecture

Input neurons pass forward input patterns to neurons in the hidden layer. In this feed-forward structure there are no connections leading from a unit to units in previous layers, nor to other units in the same layer nor units one layer ahead.

Every neuron in each layer hence communicates with only the neurons in the immediately following layer. All processing is done exclusively in the hidden and output layers. The functions performed in these layers are input function, activation function and an output function.

Input Function

The input function sums the inputs and synaptic weights. It is a linear function given by:

$$net_j = \sum_j x_i w_{ji} \qquad \textbf{Equation 3.8-1}$$

where net_j = weighted summed input to neuron j,

x_i = input i to neuron j,

w_{ji} = weight connecting input neuron i to hidden layer neuron j.

Activation Function

The most commonly used activation function for BP networks is the non-linear sigmoidal logistic function given by the following equation:

$$f(net_j) = \frac{1}{1+\exp(-net_j)} \qquad \textbf{Equation 3.8-2}$$

Output Function

The purpose of the output function is to pass the forward the output of the activation function; hence it is a linear function equal to the output of the activation function.

Processing stages

Training BP networks occurs in two stages; initially the input pattern generates a forward flow of activation from the input to the output layer. Secondly, error in the network output generates a flow of information from the output layer backward to the input layer.[47]

The back propagation procedure uses a gradient descent method, which adjusts the weight in its original and simplest form by an amount proportional to the partial derivative of the error function with respect to the given weight.[70]

The associated error for a given input pattern is calculated after the forward propagation is complete. This is done by comparing the real number output of the BP network with a target value supplied with each input patten and using the error to

update the interconnection weights from the hidden layer to the output layer. An error value is calculated for all neurons in the hidden layer prior to the output layer and similarly for each of the subsequent layers. This process continues until all weights have been updated in this manner.

Error value for output layer

The first layer error value is simply calculated as follows: [69]

$$\delta_k = (t_k - a_k) f'(net_j) \qquad \textbf{Equation 3.8-3}$$

Where, t_k = target value for unit k

a_k = output value for unit k

$f'(x)$ = derivative of the sigmoid function, and

net_j = weighted sum of inputs to hidden layer neuron j

The expression $(t_k - a_k)$ represents the difference between the target output and the network prediction while the derivative of the sigmoid function is used to scale the error.[24] The use of the derivative of the sigmoid function means that the error is scaled to make a larger correction when the weighted sum of the inputs is small, close to zero, and a smaller correction when the weighted sum of the inputs is large.

Error values for Hidden layers

The calculation of error values for the hidden layers is slightly more complicated. For neuron j in the hidden layer the error value calculation considers the weighted sum of the δ values of all neurons that receive output from neuron j. Hence the error value calculation for the hidden layer is written as [69]:

$$\delta_j = \left[\sum_k \delta_k w_{kj}\right] f'(net_j) \qquad \textbf{Equation 3.8-4}$$

where, w_{kj} = weight connection to neuron k from neuron j.

The respective values of δ are now used to adjust the interconnections of the output and hidden layer neurons. So each interconnection weight is adjusted by considering the δ value of the neuron that receives input from that interconnection. Hence this weight adjustment can be written: [69]

$$\Delta w_{ji} = \eta \delta_j a_j \qquad \textbf{Equation 3.8-5}$$

where w_{ji} = weight of connection to neuron j from neuron i, and

η = learning rate constant, $0 < \eta < 1$,

a_j = output of hidden layer neuron j

Clearly the change in connection weight is proportional to the error value, so it follows that a large error value from neuron j will result in a large adjustment to its incoming weights. Likewise, large output values, a_j, will result in larger weight adjustments. The learning rate, η, is selected to reflect the desired convergence speed of the neural network. Unfortunately very large values of η can lead to instability of the network which results in unsatisfactory learning. Conversely very small values of η will incur excessively slow learning rates. In some cases the learning rate is varied to produce a more efficient training technique for the network, perhaps decreasing in rate as the network approaches convergence.[69]

Convergence improvement

A common method for improving convergence of the weight update is the introduction of a momentum term. It is sometimes known as the 'generalised delta rule':

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i' + \alpha(w_{ij}(t) - w_{ij}(t-1)) \qquad \textbf{Equation 3.8-6}$$

where $0 < \alpha < 1$.

The addition of this term delays a portion of the weight update until the following iteration, thus oscillations in weights changes are dampened and convergence improved.

Initialisation of weights

Typically weights of a network to be trained are initialised to small random values. In fact, a well-known initialisation method for a feed-forward network with sigmoidal units is to select its weights with uniform probability from an interval $[-\alpha,\alpha]$.[73] Commonly chosen values for α are 0.5 or 1.0.

Weight initialisation has significant influence over network convergence and so is an important aspect. Training difficulties will be encountered if the network requires

unequal weights, and a uniform initialisation is used. In the extreme case, if each neuron output within the network is the same value due to equal valued weights, each weight change will be identical and the network weights will never differ. This is counter-productive, as most applications require uneven weights within the network.

3.8.1.1 Training Algorithm Summary

The training algorithm is an iterative gradient algorithm designed to minimise the mean square error between the actual output of the multi-layer feed-forward Perceptron and the desired output. It requires continuous differentiable non-linearity. For this purpose the sigmoid logistic function is generally used.[74]

Step 1. Initial Weights and Offsets

Set all weights and node offsets to small random values.

Step 2. Present Input and Desired Outputs

Present a continuous valued input vector $x_0, x_1, \ldots x_{N-1}$ and specify the desired outputs $d_0, d_1, \ldots d_{M-1}$. If the net is used as a classifier then all outputs are typically set to zero except for that corresponding to the class the input is from. That desired output is 1. The input could be new on each trial or samples from a training set could be presented cyclically until weights stabilise.

Step 3. Calculate Actual Outputs

Use the sigmoid non-linearity and Perceptron weight output equations to calculate outputs $y_0, y_1, \ldots y_{M-1}$.

Step 4. Adapt Weights

Use a recursive algorithm starting at the output nodes and working back to the first hidden layer. Adjust weights by

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x'_i \qquad \textbf{Equation 3.8-7}$$

Where $w_{ij}(t)$ is the weight from hidden node i or from an input node j at time t, x'_i is either the output of node i or is an input, η is a gain term, and δ_j is an error term for node j. If node j is an output node, then

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \qquad \textbf{Equation 3.8-8}$$

where d_j is the desired output of node j and y_j is the actual output.

If node j is an internal hidden node, then

$$\delta_j = x'_j(1-x'_j)\sum_k \delta_k w_{jk} \qquad \textbf{Equation 3.8-9}$$

where k is over all nodes in the layers above node j. Internal node thresholds are adapted in a similar manner by assuming they are connection weights on links from auxiliary constant-valued inputs. Convergence is sometimes faster as a momentum term is added and weight changes are smoothed by

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_j x'_i + \alpha(w_{ij}(t) - w_{ij}(t-1)) \qquad \textbf{Equation 3.8-10}$$

where $0 < \alpha < 1$.

Step 5. Repeat by going to Step 2.

The detailed listing of code is included in Appendix A-1: Back Propagation Source Code. The application of the BP network to prediction of roll over parameters will be covered in chapter 6 of this thesis along with the application of Radial Basis Functions, the theoretical basis of which will now be discussed.

3.8.2 Radial Basis Function Neural Network (RBF)

Neural networks are based on localised basis functions and iterative function approximation are usually referred to as Radial Basis Functions (RBF).[75] Whilst their history dates back as far as Bashkirov et al. in 1964 [76], RBF networks were introduced by Broomhead and Lowe in 1988 [77]. Early contributions were also made by Moody and Darkin [78], Poggio and Gorosi [79]. With an application approach further developed by Renals [80].

The classic characteristic of the RBF network is that the response of the localised basis function falls off rapidly as the distance between the centre of the basis function and the input vector gets large.[75] The distance scale, the function centre and the exact shape of the radial function are parameters of the model. RBF networks have been shown to approximate continuous function mapping arbitrarily well [77], [78], [80] and with the best approximation property.[80]

The RBF network is comprised of three layers, the input, hidden and output layers. The main adjustable parameters are the final layer weights, w_{kj} connecting the k-th

output node to the j-th hidden layer node. In addition to these weights there are also weights connected to all input, hidden and output nodes. There are no connections between adjacent nodes and none between non-adjacent layers as shown in Figure 3.8-2.

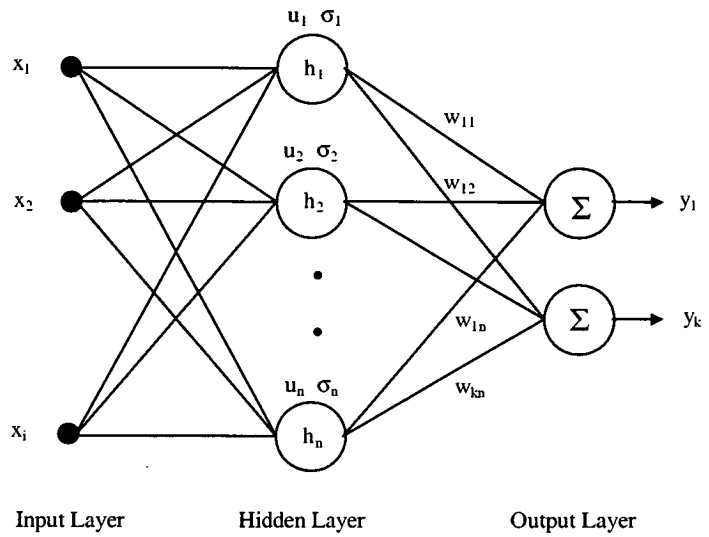


Figure 3.8-2 Radial Basis Function Network Architecture

Both input and output layers of the network consist of linear functions, as their only purpose is to pass the input pattern and output response to the external environment. The weighted summation used is generally of the form

$$y = \sum_j h_j w_{jk} \qquad \textbf{Equation 3.8-11}$$

where h_j = output of hidden layer neuron j, and

w_{kj} =output layer weight.

The number of input and output nodes is determined by the input pattern and the required number of outputs, the dimension of each vector giving the required number of nodes. As the function of the hidden nodes is somewhat non-linear, the number of hidden nodes is a more complex determination. Generally an optimum number of hidden nodes is found through a system of trial and error.

Typically the Gaussian function is used for the RBF activation function as follows.[81]

$$h_j = \exp\left[-\frac{(x-u_j)^T(x-u_j)}{2\sigma_j^2}\right] \qquad \textbf{Equation 3.8-12}$$

where h_j is the output of hidden layer neuron j,

x is the input vector,

u_j is the weight of hidden layer neuron j,

T indicates vector transpose,

σ_j specifies diameter of receptive field of hidden layer neuron j.

As may be seen in Figure 3.8-3, the Gaussian function monotonically decreases with distance from the central point. This is the classic characteristic of the RBF network.

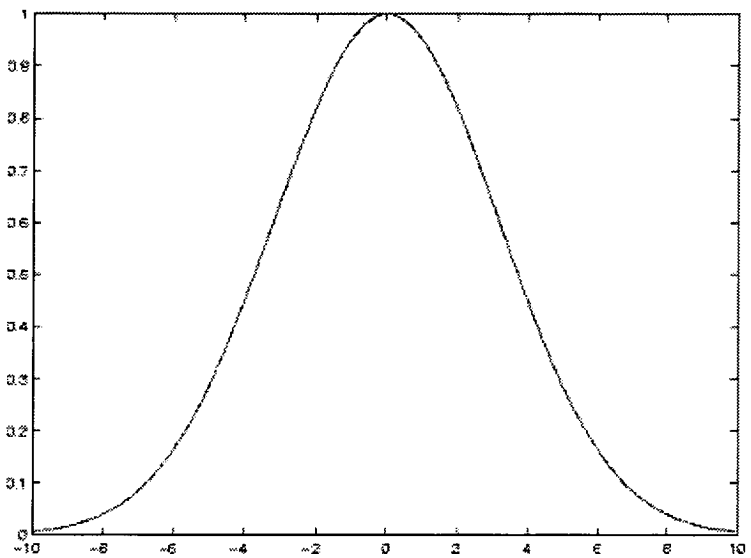


Figure 3.8-3 Graphical representation of Gaussian function [82]

3.8.2.1 Training Algorithm Summary

The simplified training procedure is written as follows:

1) Use a suitable clustering technique to set the input to hidden layer weights of the network to represent sufficiently the training patterns. Initialise the hidden-to-output layer weights of the network at small random values.[24]
2) Start the learning cycle by exposing the network to a certain input pattern paired with the desired output.
3) Compute the network's output and compare it with the desired output so that the error can be calculated.

4) Adjust the weights of the network using the error back propagation algorithm so that a certain amount of the detected error is removed.[70]

The detailed listing of the software source code is included in Appendix A-2: Radial Basis Function Source Code. Chapter 6 details the application of this network and selection of the optimum architecture.

3.9 Concluding Remarks

In this chapter covered a brief outline of the theoretical basis for neural networks as they will be applied later in this thesis. Some definitions of artificial intelligence were considered and the biological basis for artificial neural networks examined. This was followed by a brief history of the development of modern neural networks and the simple two-layer Perceptron with associated functions. The different types of networks were discussed as well as some important characteristics that make the neural network approach unique. Some considerations for improving network performance were outlined and the details of two main models discussed. These models were Back Propagation and Radial Basis Function, the source codes of which are included in Appendix A. Chapter 4 will outline the development of the experimental set up, with Chapter 5 focussing on the sensors.

Chapter 4 Development of experimental rig for training data

4.1 Design of the Intelligent Race Car

The intelligent car addresses some of the inadequacies highlighted in previous chapters. Briefly, the concept is a car that mimics intelligent behaviour, incorporating comprehensive control systems designed for maximum control and safety. The use of neural networks in this process allows not only for the simple prediction of otherwise complex relationships but also portability of the system from vehicle to vehicle once tested. The data gathered from the vehicle is used to train it. While there will be no need for extra programming for every simple change, a different training set and the results are used to customise the vehicle in question. It is anticipated that the technology will be applied to the problem of truck over turning in the future. This is a commercially viable option given the falling price of sensor technology.

4.1.1 Design considerations

In choosing a test vehicle for this project, the factors held to be of greatest importance were safety, ability to push the driving envelope and optimum sensor location. A lower centre of gravity was chosen as the preferred option to avoid physical overturning and a roll hoop bar included in the design for safety. The roll hoop bar as shown in Figure 4.1-1 was finally chosen to be round mild steel tube 25.4mm outside diameter and 2.4 mm wall thickness.

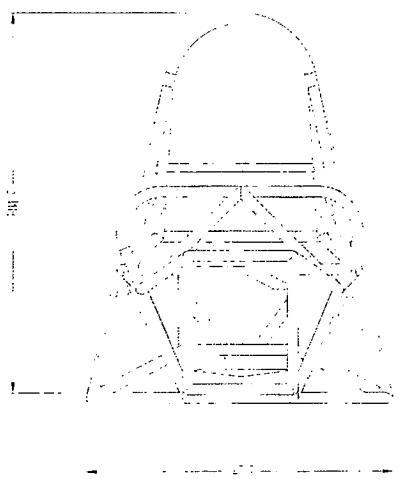


Figure 4.1-1 Roll Hoop Safety Bar.

While the car cannot physically overturn unless under extreme driving conditions, the parameters highlighted as responsible for overturning can be measured for subsequent neural network modelling. Also as this research is university based there has been a clear advantage in including an educational training component to the design.[83] For these reasons the vehicle was designed to specifications for entry into the Formula SAE, a student engineering design and racing competition held annually by the Society of Automotive Engineers in the US, UK, Europe and for the past two years in Australia. A brief outline of the race rules and design specifications is given below.

4.1.2 Formula SAE Rules

The task involved the students assuming that a manufacturing firm has engaged them to produce a prototype car for evaluation as a production item. The intended market is the non-professional weekend auto-cross racer. The car must, therefore, have high performance in terms of its acceleration, braking, and handling qualities. The car must be low in cost, easy to maintain, and reliable. The car's marketability is enhanced by other factors such as aesthetics, comfort and use of common parts. The manufacturing firm is planning to produce four cars per day for a limited production run and the prototype vehicle should actually cost below $30,000.[84]

4.1.3 Design and Manufacture

The entire vehicle was designed and manufactured 'in house' cooperatively by all team members and members of the team were involved in most aspects of the vehicle fabrication. Individual members were assigned to management of particular areas. This work focuses largely on the modifications to the engine and in the complete development of the vehicle wiring system. An overview of the entire process is included with the specific details. A separate chapter is included on the sensors as they are of particular interest.

Being the single largest component of the vehicle, design of the frame incorporated the specification of, and allowances for, all the other components. As high performance was a design goal, the frame was manufactured to close tolerances. The frame is also the primary form of driver protection where safety guidelines were included as part of the rules [84] and the minimum frame thickness for safety was

specified. The key components that were considered in the frame design are as follows:

- Suspension wishbones, springs, pull rod pivots and their mountings
- Driver interface – steering wheel, brake and accelerator pedals
- Bodywork, Seat and associated Kevlar arrangement
- Gear shifter mechanism
- Differential and drive shafts
- Engine and electrical systems
- Fuel tank
- Radiator
- Sensors

Following the initial design a Finite Element Analysis was completed [85] using the commercial package Strand7 [86]. As a result, and with consideration of material restrictions and availability, 25 mm OD mild steel tubing of wall thickness 2.6 mm and 1.6 mm were chosen for the construction of the frame. The design went through an iterative process to decrease weight whilst maintaining stiffness and strength. the end result was a frame model that was well below its strength limit and had an FEA torsional stiffness of 0.6 degrees per 1 kNm load between front and rear suspension points.

As the FEA modelling program does not feature a CAD drawing facility the model was imported into drawing package CadKey [87] as a wire frame completed in three dimensions and printed. A Complete set of drawings is included in Appendix B - Frame Specifications.

The frame was MIG welded, due to workshop constraints, in a jig that took the form of a steel table to reduce distortion. The front and rear roll hoops were bent into shape and members cut to length and tapered. Firstly, the front bulkhead was constructed followed by the engine bay and rear section, the frame was removed from the jig and final welding completed. 140 or so mounting points for the suspension, engine, seat

and harness were designed, constructed and welded to the frame. Secondly the frame was painted using automotive acrylic paint.

The next stage of construction was the suspension. The purpose of the suspension system is to maximise the amount of contact between the tyres and the road surface, in such a way as to provide the level of traction required in a situation. The suspension was designed and constructed with the following objectives in mind:

- Conform to Formula SAE rules, specifying the minimum allowable travel of all four wheels and general suspension system requirements.
- Minimise the forces experienced by the frame as all major frame loading is induced by the suspension.
- Conform to the chosen optimal suspension geometry and an appropriate level of strength and serviceability.
- Minimise the overall size (in particular the frontal area) and weight of the suspension system.
- Facilitate easy geometric changes, part replacement and tuning.
- Use as few parts as possible.
- Share as many common parts between wheels as possible.
- Provide stable mounting points for the required sensors.

For maximum design modification flexibility, conventional four wheel double wishbone suspension was used. As different geometries clearly affect the vehicle's performance under different conditions, the likely track set up for Formula SAE was considered, and a focus on maximising cornering speed was adopted. To this end the outside wheel must remain as close to vertical as possible during cornering. Desirable roll characteristics and acceptable geometry in squat/ dive under longitudinal acceleration were found from the empirical experience of the local racing community and iterative CAD modelling. The lower wishbones were chosen to meet in the centre of the vehicle to minimise width of the frame (and frontal area) and still maintain the necessary length difference between upper and lower wishbones (Figure 4.1-2).

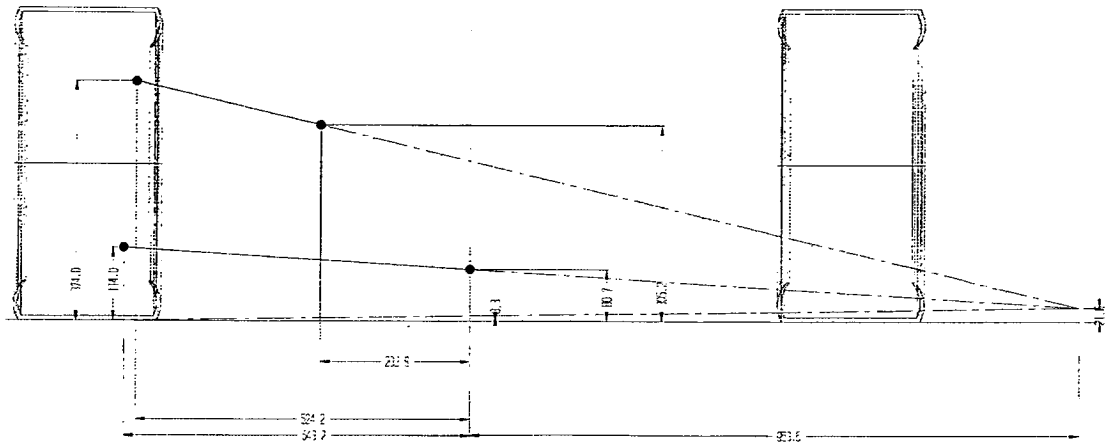


Figure 4.1-2 Front Suspension Layout and Dimensions

The heights of front and rear roll centres (the points in the transverse plane above the wheel contact patches about which the sprung mass of the car will rotate under any disturbing force) were chosen to correspond to the expected mass distribution of the vehicle. The front roll centre was chosen at ground level and the rear centre was placed at 50mm above ground level. Finally, the effective swing arm lengths were decided upon at between 100% and 150% of the track width using 6.5/19.5 –13" and 7.2/20.0 – 13" Avon racing slick tyres for the front and rear respectively as shown in Table 4.1-1. Drawings of the final suspension geometries and suspension spring specifications are included in Appendix C - Suspension Specification.

Table 4.1-1 Wishbone design parameters.

Front roll centre height	8 mm
Rear roll centre height	49 mm
Front effective swing arm length	1564 mm
Rear effective swing arm length	1114 mm
Front upper/lower wishbone length ratio	0.56
Rear upper/lower wishbone length ratio	0.71

The results of the FEA model were considered and based on this the construction material chosen to be: high strength chrome moly tubing of size ¾" diameter and 1.47 mm wall thickness for the top wish bone, 1" diameter and 2.1 mm wall thickness chrome moly tubing for the bottom. Teflon lined rod ends connected the wishbones to the frame and wheels to reduce weight. Further weight reduction was achieved through the use of a pull rod system. The steering box on the other hand is a worm

and sector steering box sourced through a drag racing company as it eliminated the need for universal joints and with a few modifications, allowed for simple adjustment of both steering wheel position and steering rate.

The final element of the rolling chassis is the wheel assembly. The wheel assembly is defined as the series of components linking the suspension and wishbones to the road. The design criteria are as follows:

- Simple and cheap to fabricate
- Light, strong and durable
- Allow for maximum braking capability
- Free rim rotation
- Mount to wishbones with adequate steering and suspension movement
- Mount for steering linkages
- Mount for wheel speed sensor
- Attachment to the drive axle for the rear wheels
- Give desired scrub radius and king pin angle
- Allow for significant adjustment in parameters such as camber and caster for drivability
- Parts commonality.

Final wheel assembly specifications are given in Appendix D.

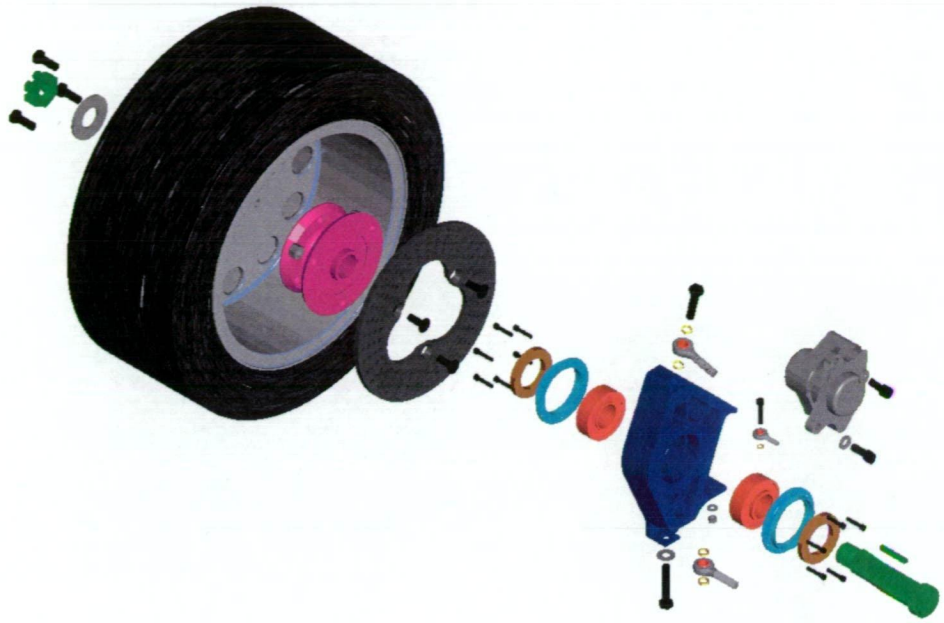


Figure 4.1-3 Exploded view of final wheel assembly.

13" rims were chosen to allow room for large reliable brakes with a trade off on added wheel mass compared to 10" rims. The rims were custom made to suit the required dimensions of 7" width, 144 mm backspace offset and 152 mm internal diameter. As previously mentioned the chosen tyres were 6.5/19.5 – 13" and 7.2/20.0-13" front and rear respectively to minimise sidewall deflections on the front tyres and aid acceleration by acting as energy absorbers on the rear. The brakes were chosen from Wilwood High performance Disc Brakes. 10.2" diameter 3 pin mounted aluminium rotors in conjunction with aluminium billet dynalite single callipers common across front and rear wheel assemblies.

After extensive analysis considering an off-centre bearing design and a centred design, the off-centre design proved to be simpler and cheaper to manufacture and was chosen at the cost of slightly lower performance.[88] The design featured a 35 mm stub axle and 35 mm ID and 72 mm OD single row tapered roller bearing owing to perceived difficulties in fabrication for marginal performance increases using the centred bearing design.

To connect these elements to the suspension the uprights were designed with the following considerations:

- Bearing and bearing spacing
- Position and strength of suspension mounts
- King pin angle, caster and camber adjustments
- Placement of the brake calliper and steering mount
- Milling limitations
- Weight reduction and appearance

The final design was CNC machined to shape and the boltholes manually drilled and tapped. Drawings are included in Appendix D.

Driver interface with the vehicle is an important aspect of a user-friendly design. Ergonomics and weight reduction were the primary concerns of the cockpit layout. The design highlighted a need to have full closeout between the driver and possible course hazards, including the road, front on and behind from the engine and other moving components. Visibility was important and a reclined seating position was chosen to minimise resistance from frontal area. The seat and nose were moulded in carbon fibre, foam, carbon fibre / Kevlar layers in a female mould custom made from plywood. The head of the seat is detachable for easy engine access.

The gear shifter is a simple pivoted shaft to the sequential motorcycle gearbox. A hand clutch is mounted on the gear lever and its movement is cable driven. The brake and accelerator were mounted on aluminium plate to allow for individual driver adjustment. The accelerator was produced from aluminium plate and the brake system was supplied by Wilwood and includes twin composite master cylinders that control individual front and rear brake circuits proportioned by a balance bar.

The drive train was designed specifically around the principles of low rotating mass and efficient power transmission. Standard CV joints were replaced in the design with the use of Kevlar composite disks with an associated weight saving of 6 kg. Power transmission efficiency is close to that of a solid rotating shaft with a torque rating of over 1000 Nm. This efficiency required alignment to within 1 degree although instantaneous deflections of up to 6 degrees may be withstood.

The drive shafts ware large diameter aluminium-alloy tubes that are both stiffer and lighter than the traditional solid steel. This design is dependent on a flanging arrangement to connect the shafts to the composite discs. FEA analysis was used to evaluate alternative designs for sections based on torque ratings and torsional deflections.

A fully sealed Quaife Automatic Torque Biasing (ATB) differential was chosen to assist maximum traction during acceleration and cornering. A blank sprocket was machined to match the bolt pattern. The differential was mounted with an asymmetrical diagonal member supporting the top of the right-hand side differential mount that also provides triangulation of the drive train frame structure. Details of the differential and composite discs are included in Appendix E – Drive train Specifications.

4.1.4 Engine and Electrical systems

The engine and electrical system are difficult to separate and so are included together. Limitations imposed by the Formula SAE guidelines [84] meant that the available engine capacity was 610 cc or less. This significantly reduced the options on the type of engine to be used with a motorcycle the most beneficial option. A 2000 Kawasaki Ninja ZX6 was chosen from a range of engines for its large cam overlap, low rpm torque and its lightweight construction. The deep oil sump is ideally suited to high lateral accelerations. The engine is a double overhead cam, four stroke four cylinder liquid cooled model with a compression ratio of 11.8:1 and 6 forward gears, the capacity is 600 cc.

In developing a working electrical system for use in the experimental test vehicle the following objectives were identified as essential:

- Open architecture and expansion capability for addition sensory input for further extension of on-board electrical systems during automotive neural network development.
- Simplicity using common parts where feasible and designing straight runs minimising the number of wires.

- Fault Diagnosis – creating fault finding access locations, common locations of like components and using fully colour-coded looms.
- Minimal maintenance requirements – the system developed must be robust and require little maintenance.
- Aesthetically neutral/appealing.

The wiring loom was designed to fit neatly away from the major moving parts of the car most likely to cause damage. For this reason the ECU (Engine Control Unit) is located well away from the engine under the seat with the ignition module bolted to the floor. As the ECU casing was waterproof further protection was not deemed necessary. The wiring loom runs down the right side of the driver, away from the gear changer on the left, to the fuse box.

Figure 4.1-4 Fuse box location.

The fuse box is located in a central position to minimise the length of wiring to and from it from all locations on the vehicle and for maintenance purposes. The bulk of the remaining loom continues along this line at a junction with the wiring from the battery located near the starter motor and alternator on the left-hand side of the vehicle. The original starter motor, alternator and regulator wires have been retained from the original motorcycle engine loom.

For reliability during testing it was imperative that the complete wiring loom be dependable in all weather conditions. The major concern in this case was rain and contact with water and mud under wet track conditions. The insulated cables within the wiring loom were further insulated by comprehensive wrapping in electrical tape in all areas exposed to the elements. The most vulnerable components of the wiring

system were the system of relays and fuses that allow the MoTeC system to control the higher currents of the engine control system. Hence the fuse box containing these sensitive components and exposed connections required special attention. The fuse box is carefully sealed to avoid difficulty in this area.

The design of the wiring loom assigned a unique wire colour to each component for simplicity of fault diagnosis. Basic electrical conventions were observed as far as practical (black for earth, white or red for power.) Striped automotive wiring would be used for production runs of the loom to ensure colour individuality. Wire utilised in this design was chosen to withstand the harsh environment of the automobile. Wiring sizes were taken from the largest likely current rating of each individual component they served. For instance, the fan wiring could expect a continuous operational current of between 2 and 3 Amp with a stall or start up current in the vicinity of 7 Amp. In this case the wire chosen was rated at 10 Amp. This was deemed necessary for the absolute reliability required of the electrical system.

The wiring was wrapped tightly and neatly by hand and covered where possible with conduit for reasons of aesthetics and extra durability.

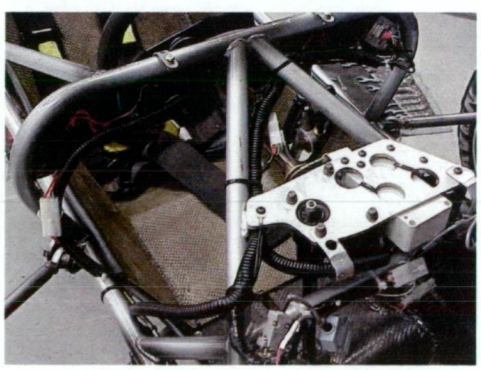


Figure 4.1-5 Conduit, triple insulated wiring loom under nose cone.

Care was taken to ensure that the wiring resides as far from hot components as possible. Heat from the engine was enough of a problem to other components of the car that heat shielding on the exhaust was deemed an appropriate solution. To ensure easy removal of all components, wires were crimped and not soldered. Where possible large multiple connectors have been used to eliminate the possibility of incorrect connection upon re-installation.

For safety reasons power for the entire car may be instantly cut using any one of three kill-switches (shown in Figure 4.1-6). The brake over-run switch is located behind the brake pedal and will activate if the brake is depressed while there is simultaneous failure of both independent braking systems. The other two kill switches are located on the dashboard to the driver's right and externally next to the head support on the roll bar to the driver's right, in accordance with best practice international FIA safety standards.

Cutting any one of these switches will deactivate the switch line to the main relay. All power to components on the vehicle is sourced from this relay or from the switch line after the kill switches. Both human operated kill switches are clearly labelled with the international electrical symbol of a red spark on a white-edged blue triangle. Circuits and wiring diagrams are included in Appendix F – Electrical Specifications.

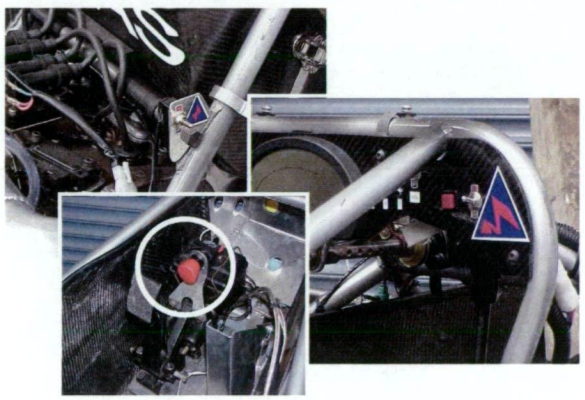


Figure 4.1-6 Kill switches

The primary purpose of the kill switches is to ensure that in case of an emergency the engine will be deactivated. This brings us to the choice of engine and the development of its support systems.

The choice of engine and the poor performance of carburetion under high lateral acceleration led to the development of a custom fuel injection system. It was envisaged that the vehicle would eventually feature installation of exhaust gas turbo charger. While this addition did not eventuate the design of the inlet and exhaust reflect this intention.

Formula SAE specified that the fuel system must have a volume not exceeding 7.5 litres and that the maximum distance raced is 22km.[84] The fuel tank was designed to be just 5 litres and placed under the seat. Space limitations meant that the tank was required to be long and shallow, problems with surge were avoided through the use of foam and a small baffled chamber at the fuel pump pick-up.

The exhaust was designed to fit the existing outlet ports on the engine. The effective lengths of the outlet pipes were modelled on an existing system and fabricated from stainless steel mandrel bends. The exhaust was fabricated from a stainless steel pipe, perforated and surrounded with fibreglass packing, to minimise backpressure. The tail pipe length was acoustically tuned to a minimum of noise output at 9000 rpm. The resulting 109 dB at 0.5 metres from the exhaust was within the limits imposed by the Formula SAE rules.[84]

The cooling system consists of a radiator, electric water pump and two thermo-electric fans. Initially, the radiator was envisioned as being placed in a side-pod on the vehicle as this would ensure more effective cooling. After much deliberation, the location was changed to the rear of the vehicle to minimise drag and the weight inherent with the extra hose lengths, also it was thought that the slow speeds of the Formula SAE event would minimise the benefit of the side-pod arrangement. The fans and water pump supply the extra fluid flow required to keep the engine cool.

The chosen engine management system was the MoTeC M4-Pro Engine Control Unit (ECU). The system provides sequential injection, which is ideal for fuel efficiency and optimum fuel injection timing. It also provides 3D-mapping of engine parameters and the option of open or closed loop operation from an exhaust oxygen (Lambda) sensor for improved fuel economy or performance control. It includes engine oil pressure and cooling water temperature sensor inputs as well as the facility to run engine based traction control based on wheel speed measurements. The system stores up to 128 kB of logged engine data which allows for a complete analysis of the engine after running. The system is shielded from radiated interference and includes filters to reject low impedance conducted interference. Essentially the ECU is a 32 bit micro-controller. Sensors are read at up to 2400 Hz and the entire program regenerates at

200 Hz. The system is said to use up to 70% less power than other systems to fire injector hence drawing less power from the electrical system and generating less heat.

In fitting with the major objectives of the electrical system and engine convenience, the original loom design with in line fuses and multiple bulky relay mountings were re-designed to incorporate a single fuse and relay box.

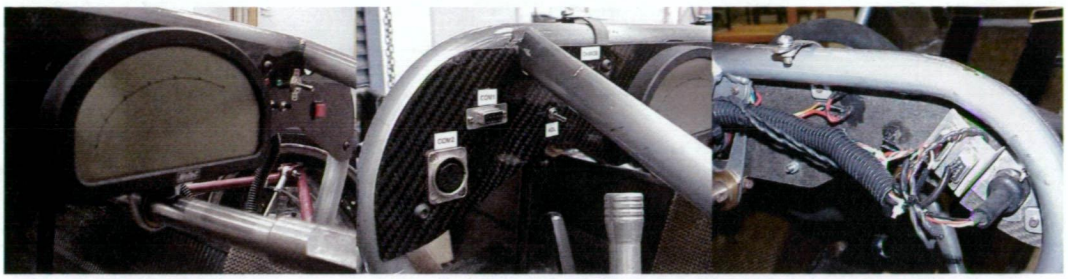
The box itself is made of clear material for reasons of fast identification of loose connections or burnt out fuses and relays as well as aesthetics (shown previously in Figure 4.1-4). The wires are fed in groups through the back of the box that allows removal of the box without the loom and vice versa. Rubber grommets have been used along with a rubber gasket to seal the box cover in place.

The circuit for the fuse box was taken almost directly from the MoTeC ECU wiring diagrams (Appendix F). Major alterations involved the removal of a relay that was redundant in this particular application and the unification of relay type used throughout the vehicle. The required diode activated relay was replaced with an in line diode on the circuit to allow fitting of a standard relay in its place. Extra fuses were included for each of the major components, allowing for very simple component fault diagnosis.

One of the major objectives of the fuse box was to find a neat and effective way to mount all the required relays and fuses in one place where they could be easily accessible to check and change. The ideal situation was chosen to be a set of closely located relay and fuse sockets joined to plugs as appropriate by a printed circuit board. The printed circuit board was layered with solder tracks to increase the current rated capacity in excess of 12 Amp (Appendix F).

Load ratings of fuses were selected according to the component they serve. In general fuses will withstand up to their rated current plus 200% for 2 minutes. The majority of components have around 7-8 Amp start up current with a continuous 2-3 Amp while running. 10 amp fuses are most common in this system (as shown previously in

Figure 4.1-4). Automotive fuses were specified for their compactness, physical durability (plastic cases) and ease of mounting.



(i) (ii) (iii)

Figure 4.1-7 (i) ADL located on dashboard, (ii) Com ports are located on the dashboard for easy access during testing and data transfer with the remote PC and (iii) Rear connections from wiring loom to Com ports.

The dashboard is the driver control panel. All the information about the engine and other aspects of the running car is fed to the driver through the Advanced Dash Logger or ADL. Information about the ADL is contained Appendix G along with reference wiring and programming.

The dash itself is constructed from a composite sandwich of carbon fibre and foam constructed using vacuum bagging techniques, the fibre was said up on sheets of window glass to ensure a good finish. Profile cutting, drilling and finishing all performed after the bonding process. The dashboard attaches to the frame through bolts and welded tabs.

COM ports featured in the dash are used to facilitate fast and easy connection of the on-board computer systems to an external computing and storage PCs for data logging sessions. The COM ports connect to plugs behind the dash where the ECU and ADL are connected facilitating easy removal of individual units. The trackside computer is simply plugged into the front of the dashboard to download data or change on-board programmed settings.

As vibration is significant on the test vehicle, stress relief for the wiring and solid mountings is a feature of this design. Ease of removal of the dash was a major

consideration and was achieved using two main connectors behind either side of the dash.

Once the physical layout of the dashboard was determined, the circuit design was largely a matter of connecting the components with the main vehicle circuit. Shrink fit wire covering was used to prevent shorting, and the back of the dash is painted with liquid electrical tape to ensure water resistance. Physical layout of the dashboard switches and lights follows a logical progression. The most important switches are to the right, away from the gear changer. The neutral and oil warning lights are located close to the ignition switch. Specifications for components of the electrical system are included in Appendix F - Electrical system general specifications.

4.2 Concluding Remarks

This chapter addressed the design and construction of the test vehicle. Safety while testing parameters that contribute to vehicle roll over was an integral part of the design. An outline of design and fabrication was given for the frame, suspension, wheel assembly, driver interface and engine systems with the electrical system covered in more detail. Chapter 5 will examine the significance and placement of the sensors throughout the vehicle. As all the results obtained are dependent on the sensor outputs the following chapter is integral.

Chapter 5 Sensors and Sensor Fusion

Sensor fusion usually refers to the combination of multiple sensor data into one representation or control action for improved measurement accuracy or motor behaviour.[89] In this case, the vehicle dynamic sensor outputs are intended for prediction of accelerations and velocities. As a preliminary step, the sensor output must be collected through the data acquisition system and later collated for the purpose intended by the various neural networks. The following chapter outlines the sensors themselves: their calibration and positioning followed by the various components of the data acquisition system. The use of the data in prediction and decision making is covered in chapter 6.

5.1 Sensor Positioning and Specification

A detailed analysis of sensor selection may be found in [90]. The most critical sensor from a positioning point of view is the acceleration sensor. Engaged to measure accelerations in 3 dimensions, yaw angles and rates, pitch angles and rates and roll angles and rates, the acceleration sensor must be located at the vehicle's centre of gravity. This is due to the way in which the sensor uses the signals it receives. To calculate actual pitch and roll angles, the angular rate signals must be integrated. Unfortunately, an offset error in angular rate will produce an error in angle. That angle error increases linearly with time. In addition, the random noise in the rate sensors will produce a random walk effect in the calculated angle. The random walk causes the calculated angle to drift at a rate proportional to the square root of time, even in the absence of rate-bias error.[91] Many of these difficulties can be avoided by initially mounting at the vehicle's centre of gravity and roll, and correct directional alignment. These considerations were incorporated in the design of the vehicle.

The following quantities were identified as integral to the dynamic performance of the vehicle:

Chassis Parameters:

- Individual Wheel Speeds, ω_w in rev/s
- Accelerations, A_x ,A_y, A_z, in m/s^2
- Yaw Angle, ψ in degrees.

- Pitch Angle, γ in degrees.
- Roll Angle, θ_r in degrees.
- Yaw Angular rate, $\dot{\psi}$ in degrees/s.
- Steering Angle, δ in degrees.
- Suspension Spring Travel, Z_{st} in m.
- Brake Hydraulic Pressure in front and rear circuits, F_B in Pa.

Engine parameters:

- Engine rpm
- Throttle position
- Lambda value

Calculated parameters:

- Distance, D in m.
- Gear, calculated from rpm and drive speed
- Pitch and roll angular rates, from the first derivative of the angles.
- Ground speed from the averaged front wheel speeds
- Distance, the integral of ground speed.
- Drive speed, averaged rear wheel speed.
- Driven wheel slip from rear wheel speed and ground speed.
- Individual wheel accelerations, the derivative of the wheel speeds
- Individual suspension velocities from differentiated suspension positions.

Mathematically, the dynamics of a vehicle may be expressed as a function of all of the above parameters. The selection of sensors to collect readings of the above parameters was limited by the following:

- Minimal available mounting space
- Ability to withstand harsh environmental conditions
- Communication and signalling compatibility between elements of instrumentation
- Sensor availability [90]

Table 5.1-1 Summary of input instrumentation sensors.[89]

Measurand	Sensor Type	Supplier	Quantity
Wheel speed	Honeywell Gear Tooth GT1 Series Hall Effect Sensor	MoTeC	4
Three axis accelerations, roll, pitch and yaw angles	Crossbow DMU AHRS400-200 Sensor	Davidson Industrial Measurement	1
Steering Angle	MoTeC (Spectrol) 10 turn, gear driven rotary potentiometer	MoTeC	1
Spring Travel	Gefram Linear Potentiometer – 100 mm	MoTeC	4
Brake Force	Honeywell Eclipse Pressure Sensor – 2000 psi	MoTeC	2

The sensors summarised in Table 5.1-1 are fundamental to the accuracy of the data required for this study and thus will be covered in detail after the engine sensors.

5.1.1 Engine Sensors

The throttle position sensor is located on the inlet butterfly valve and provides the ECU with a variable voltage that represents the position of the throttle. This information is then used to control air-fuel ratio, timing and fuel shut-off.[92] Calibration is of the fully open position and fully closed position through the software associated with the ECU. After comparison with the Manifold Air Pressure (MAP) sensor, the throttle sensor was found to give better response after the engine was tuned.

The standard crank angle sensor on the ZX-6 engine was kept and used as an input to the ECU. The sensor is a Hall Effect sensor triggered by a wheel with twelve evenly spaced teeth. To fix a point in the rotation cycle, the twelfth tooth is in actuality missing. Thus the gap in the signal corresponds to pistons two and three being at top dead centre whilst one and four are at bottom dead centre. In a four-stroke cycle this is not sufficient information to determine the position of each piston in the cycle. However, the decision to run wasted spark meant that the spark plugs each fire at the top of the piston stroke regardless of whether it be the compression stroke or the

exhaust stroke. This type of ignition system is common on standard fuel injected engines.

Sequential fuel injection is the most precise way to inject fuel into an internal combustion engine. Other methods of injection involve injecting some or all of the cylinders at the same time, this is undesirable as the fuel may not fully atomise due to the stationary air and evaporation into the inlet manifold can occur due to the heat of the inlet valve. Sequential injection means that each injector fires individually at the correct moment in the cycle. As the spark plugs were running wasted spark a reference point was needed to identify the current phase of each piston in the cycle. To achieve this an additional sensor was required on the camshaft as this rotates only once per four-stroke cycle. A common Hall Effect sensor and associated wiring was mounted to a stainless steel bracket inside the engine and triggered by a remote earth magnet fixed to an existing hole in the cam shaft timing sprocket. The wires were insulated using a Teflon insulator and exit the engine through a hole drilled in the rocker cover.

The Hall Effect works as a proximity switch under the influence of a magnetic field. The use of the stronger remote earth magnet allowed the distance between the magnet and the sensor to be increased from 2 mm to 5 mm, for convenience of mounting. The device is rate up to 100 kHz repetition rate.[93]

To be read the pulse from the sensor must be amplified by the ECU input. A simple MOSFET amplification circuit is used for the job. The circuit is built on printed track circuit board and mounted in a box on the outside of the engine. The sensor wiring is temperature resistant; Kevlar insulated wire designed for harsh conditions. The sensor is mounted into the cam sprocket cavity in an aluminium bracket held by the nearest cam cover bolt. Directions for the set up and calibration of this sensor are included in the ECU help menus.

The Lambda sensor measures the oxygen content in the exhaust gases, which is directly related to the air/fuel ratio of the burned fuel in the combustion chamber. It is mounted on the outlet engine pipes as close as physically possible to the engine to

minimise the time delay between the lambda reading and the engine rpm reading. A Zirconia cell generates the voltage output of the Lambda sensor. In principle, a process gas with unknown oxygen concentration flows over a measuring probe that is sealed off from the process gas by the heated Zirconia cell. A reference gas on the opposite side of the zirconia cell with its known oxygen concentration contacts the cell from the inside surface. At high temperatures, a voltage is generated between the two surfaces of the cell. At constant cell temperatures this voltage depends only on the ratio of oxygen concentrations between the reference gas and the process gas.[94] As this voltage is temperature dependent, the sensor is temperature compensated based on the increase of resistance of the sensor cell with temperature.

5.1.2 Wheel Speed Sensors

The wheel speed sensors are digital output gear tooth Hall Effect sensors. The Hall Effect is an electrical phenomenon discovered in 1870 by Dr Edwin Hall. When a current flows through a conducting material a magnetic field is set up. It means that a voltage is generated transversely to the current flow direction in an electric conductor (the Hall voltage), if a magnetic field is applied perpendicularly to the conductor.[95] If a ferrous material then passes through this field it concentrates the magnetic flux away from the conductor causing change in voltage. By measuring the changing voltage the passage of conducting material through the magnetic field may be monitored. In this particular situation the movement of the ferrous material, ie the bolts holding the wheel inners to the wheel rims, gives a changing voltage indicative of the wheel rotation.

The Hall Effect sensor is composed of an integrated circuit made up of discrete capacitors and a bias magnet sealed in a probe type, non-magnetic plastic package for physical protection and cost effective installation. Wiring consists of a power source or voltage, sensor ground and signal wire that provides the output or measurement reading value. The sensor uses a discrete capacitor to store a reference voltage that is directly proportional to the maximum magnetic field strength (ie. the absence of a wheel bolt). A digital output signal is triggered when the magnetic field sensed by the hall element changes by a predefined amount. A feedback circuit is integrated into the silicon circuit and used to reduce the effects of temperature and other error inducing variables.

Hall effect sensors generally operate using a metallic gear tooth wheel that has the function of concentrating the magnetic flux away from the sensor. Incorporating such a gear into all four wheel assemblies proved a difficult design task and as such an alternative solution was found. As previously mentioned the heads of the steel bolts, that hold the aluminium face plate of the wheels to the rims, act as the sensor target material with the aluminium face plate acting as the non-target material. Simple mounts were constructed to support the sensor and mounted to the brake calliper.

Calibration of the sensors proceeded once they were wired to the ADL. One wheel rotation is equivalent to 18 Hall effect pulses; the wheel speed calibration was effected in the software by specifying this value and the measurements of the individual wheel circumferences.

5.1.3 Acceleration Sensor

Based on the difficulty of positioning a cluster of sensors at the vehicle's centre of gravity a single unit was chosen with the capability to read all the variables required from the one position. The DMU-AHRS is a sensor clustering measurement system, designed to measure nine parameters including stabilised pitch, roll and yaw angles and yaw angular rates, and acceleration about 3 axes. These values are measured by using a combination of micro-machined three axis accelerometers, three axis rotational rate sensors, and three axis magnetometers. The addition of the three axis magnetometers allows the unit to make a true measurement of magnetic heading. Output may be in analog or RS232 digital form. The unit uses a combination of different methods, the principles behind which are detailed below.

Firstly an accelerometer works by measuring the relative displacement of a spring mass system under acceleration. In the case of this particular sensor, three micro-machined silicon micro electrical mechanical system (MEMS) accelerometers use differential capacitance to sense acceleration. This type of accelerometer senses a change in electrical capacitance through the use of a distorting diaphragm sandwiched between two plates. The two plates form the capacitor unit and detect changes as they are separated due to the movement of the diaphragm under acceleration in one plane.

The sensor then conditions the signal to create a linear output, as capacitance differentials are not directly linear.

Next, the three angular rate gyroscopic sensors are made up of a number of vibrating ceramic plates that use a silicon MEMS structure to measure the Coriolis force ($F=2m\ \omega V\sin(\phi)$, where ω is the angular velocity of the axis, V is the relative velocity and ϕ is the angle between vectors ω and ϕ) induced by the dynamic movement of the test apparatus. This data may then be used to calculate the rotation rate around the given axis. The advantage of this approach is that the output angular rate is independent of the acceleration output. One significant problem that arises is that a change in direction around one axis of a driving transducer induces a vibration in the detection transducer on another axis. To over come this problem an oscillator circuit is used to control the vibration.

Finally the three magnetometers within the unit are constructed as miniature fluxgate sensors and are used to provide the heading angles with respect to the earth's magnetic field. These results are used only for reference, with the angular rate integrals and gravity angles used to stabilise the results.

The acceleration sensor needed to be mounted as close to the centre of gravity as possible to minimise measurement errors as the sensor measures acceleration proportional to the product of the angular rate squared and the distance to actual centre of gravity. The required location was found from the FEA conducted in the design phase and verified through the measurement of the vehicle's weight distribution at each wheel. Vertically the only available option was to mount the sensor on the floor behind the driver. The fuel tank was designed to accommodate this location.

Installation involved taking particular care to avoid ferrous materials close to the sensor that could affect the accuracy of the magnetometer. The sensor is mounted on a layer of foam to minimise vibration and held in place with velcro strips to add further dampening and eliminate the need for bolts. Connection proceeded through the ADL

using the analog input channels and calibration was done within the software by setting the acceleration due to gravity of the sensor upright, on its side and on its back to give the three directions.

5.1.4 Steering Angle and Spring Travel Potentiometers

The principle of operation for both the steering angle and spring travel sensors is the same so they are included together. Both sensors are potentiometers, the suspension travel sensors are linear and the steering angle sensor is rotary. A potentiometer is an analog sensor; it operates on the principle that electrical resistance is proportional to resistance length (in a straight line or around a curve). They generally consist of a movable component that makes contact at a point along an internal resistance. Thus the current flowing through the circuit encounters more or less resistance based on the position of the sensor element and the voltage changes accordingly, this is the sensor output.

These sensors consist of an anodised aluminium cylindrical case with an internal moveable control rod of stainless steel. The maximum possible extension is 100 mm. The sensors have M5 self-aligning rod ends at each end for mounting. The linearity accuracy is 0.05% with infinite resolution based on the analog output.

Installation of the linear potentiometers required sufficient length to be left in both upward and downward directions of suspension travel to avoid damage of the sensors travelling over small bumps and ditches. Using a calibration technique whereby a known deflection is applied at each individual wheel and associated with the sensor output voltage in the software, the sensors were able to be mounted with no requirement of perfect alignment and orientation precision in the vertical plane. This simplified the mounting procedure enormously. Eight mounts were welded to the frame and wishbones of the vehicle and the sensors bolted into position with washers to allow the rose joints at each end to self-align during operation. The sensors were zeroed with the car standing on the ground with an average sized driver.

The steering angle sensor incorporates a large pulley that was mounted directly to the steering wheel shaft and the actual steering angle sensor mounted in the direct vicinity using an "L" shaped channel of aluminium. The sensor could be calibrated to measure either the actual change in the steered wheel angle of the vehicle, or the angle of the driver's steering wheel. In this instance the latter was chosen. The calibration consisted of temporarily mounting a protractor to the steering wheel and taking a voltage reading. The steering wheel was then moved through a number of known angles and the voltages read. From these values a calibration curve was developed and entered into the software.

5.1.5 Brake Force Pressure Transducers

The brake pressure sensor works on the piezoelectric effect. This effect occurs when an external force strains a crystalline substance such as quartz, resulting in a measurable charge accumulation on the crystal surface as its ions are displaced. The electrical voltage that develops across the crystal due to mechanical displacement is also proportional to the input pressure that causes the deformation of the crystal and forms the basis of pressure transducer operation. The two pressure transducers were installed into the brake lines through the use of a simple T-junction arrangement close to the nose of the vehicle for ease of access and wired into the ADL loom. Calibration tables were included in the Dash manager software and calibration was thus simply a process of loading the appropriate file into the system.

5.2 Data Acquisition Instrumentation

Data acquisition instrumentation refers to the equipment that processes the sensor voltage outputs into meaningful information on the vehicle, the equipment that transmits this to and receives the information at the research computer off the track and the software that is then used to manipulate it. The following section outlines the equipment chosen including issues of installation.

5.2.1 MoTeC Advanced Dash Logger (ADL)

The Advanced Dash Logger (ADL) from MoTeC is a compact complete data acquisition system suited to on-board collection of sensor outputs and engine control

parameters from the ECU. It also serves as a display unit to give the driver feedback from the vehicle systems.

Specifications:

4 Mb data storage memory capacity allowing 16 minutes of testing

10 analog inputs

8 digital inputs

RS232 serial input

4 auxiliary outputs

High speed 32 bit microprocessor

Programmable high contrast Liquid Crystal Display

Detachable wiring loom.

CAN communication cable for direct PC connection

Installation of the ADL was largely a process of optimising visibility for the driver making the CAN cable easily accessible on the dashboard and connecting the wiring loom appropriately. A separate 12-volt battery was included as the power source to isolate any interference from the heavy electric system associated with the engine.

5.2.2 Real Time Clocks

The installation of a real-time clock provides time and date channels to the ADL to allow time stamping of all measured data. This unit also provided an additional RS232 input communication port for the ADL.

5.2.3 Telemetry

The telemetry system consists of two modems, complete with computer connections and antennas. The transceiver modem is located on the vehicle and transmits data logged by the ADL to the transceiver model that is connected to the data storage and display computer located off the track. Both modems operate in the 900MHz-frequency band and utilise a pseudo-random code that enables the transceiver modem to transmit data on various frequencies throughout the band. This minimises interference from other sources.

5.2.4 Software

A number of software utilities were included in the purchase of the data acquisition system. These included: MoTeC Dash monitor for programming and modifying the dash display, Data interpreter software which includes an export option to allow use of data in other packages, such as Excel. The software for programming the ECU was also included in the purchase. These software tools allow for the calibration of sensors and the manipulation and display of data during and after collection.

5.3 Data Accuracy and Sensitivity

The final accuracy and reliability of the trained neural networks depends heavily on the accuracy of the input data. This accuracy is dependent on the sensor accuracy as well as the precision of the entire measuring system in transforming the analog and digital signals to true units. The transmission and storage of data also plays a part in this. For example, a low sampling rate of a highly variable data set can result in a loss of information, as the true variation is not represented in the sample. Clearly this aspect of data acquisition is open to optimisation.

5.4 Testing and performance

From a design point of view the test vehicle was itself thoroughly tested along with the telemetry system as part of the development process. It proved to be extremely reliable and robust. The final test was racing in the local Formula SAE where it performed well and held up flawlessly through the rigorous dynamic testing. The vehicle has done over 1000kms without significant failure.

5.5 Concluding Remarks

This chapter discussed the sensors and sensor fusion considering sensor positioning. The key sensor was shown to be the acceleration and angle sensor that was mounted as close to the vehicle centre of gravity as possible. The broad list of parameters to be measured was identified and the various sensors described and specified, beginning with the engine sensors followed by the dynamic sensors. Installation and calibration of the sensors was also included. Finally, the data acquisition system was specified

including telemetry, real-time clocks and software for on-line parameter estimation. The results of research testing, data acquisition and network training are the topic of chapter 6, to follow.

Chapter 6 Prediction of Parameters to Avoid Roll Over.

A comprehensive range of testing parameters for varied testing conditions has been examined. This chapter details the predictions for the roll over parameters. Two neural network architectures were considered Back-propagation and Radial Basis Function. A comparison of these predictive models was made for the cases of velocity and roll angle prediction as important parameters for the prevention of vehicle roll over. A rationale for the choice of variables will now be addressed followed by details of the selection of most appropriate network architecture.

Two parameters were chosen for estimation toward the prevention of vehicle roll over. These were based primarily on a fundamental analysis of the forces acting on the vehicle. The analysis is as follows:

Consider a vehicle of mass m, travelling a curve of radius R to the vehicle centre of gravity at a longitudinal velocity v.

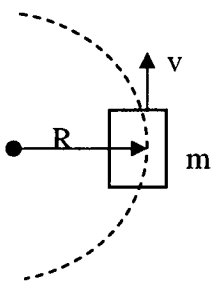


Figure 6-1 Vehicle mass, radius and velocity

Centripetal Force is given by:

$$F_C = \frac{mv^2}{R}$$ **Equation 6-1**

For a vehicle with stiff suspension, the roll over moment about point A is given by:

$$M = \frac{mv^2}{R} \times h$$ **Equation 6-2**

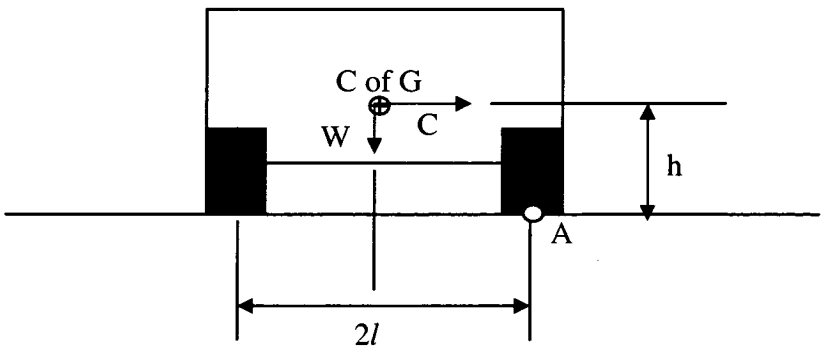


Figure 6-2 Vehicle Moment and Force Diagram

The opposing moment about A due to weight is given by:

$$W = mgl$$ **Equation 6-3**

Hence the velocity limit for a stiffly sprung vehicle where:

$$\frac{mv^2}{R} \times h = m\lg$$

is given by $v = \sqrt{\frac{glR}{h}}$. **Equation 6-4**

However, for a vehicle with suspension, the vehicle body-roll, as indicated by the vehicle roll angle θ_r will vary the length of both h and l.

Clearly the two critical parameters are roll angle, θ_r and vehicle longitudinal velocity, v. The prediction of these values is to be done based on the sensor outputs from the test vehicle. In order to train a neural network it is necessary to have measured the correct value in some way. In the case of θ_r, this is determined directly by a gyroscope system in the acceleration sensor (outlined in chapter 5). A true measure of the vehicle velocity is somewhat more complicated as wheel speeds can differ from the vehicle speed in cases of wheel lockup or when a wheel leaves the ground in tight cornering. To alleviate the impact this may have on the results, the method of Porcel et al.[13] was used to determine a close approximation. The use of this method is based on the results given by this research team and outlined in chapter 2. The estimation of these values was from non-wheel speed sensors only and shows the validity of such an approach should an expensive optical sensor be used for training

purposes. This application of the approach is outlined in the following section. The appropriate tool to predict these two parameters demands selection of neural network architecture.

6.1 Selection of Appropriate Architecture

The effect of changing architecture affects performance from network to network. In the case of a robust network such as the 2-layer Back Propagation network the dangers are mainly those of over or under-training and excessive time consumption. The following results were obtained through a numerical investigation over the range of 2-10 hidden first and second layer nodes for the BP network and 2-10 nodes and sigma between 0.1 and 0.5 for the RBF network as shown in Table 6.1-1.

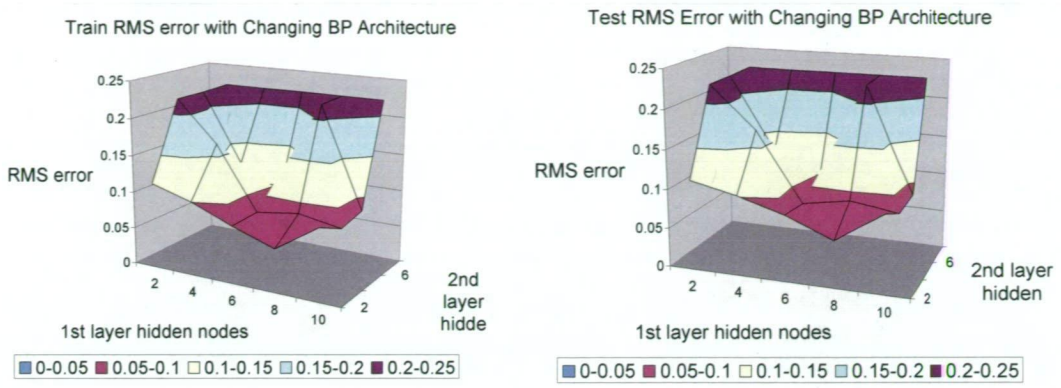
Table 6.1-1 Range of Numerical Investigation Covered.

BP		RBF	
Inputs	16	Inputs	16
Output	1	Outputs	1
No. Hidden nodes 1st layer	2-10	No. Hidden nodes 1st layer	2-10
Transfer function	Sigmoid	Sigma	0.1- 0.5
No. Hidden nodes 2nd layer	2-10		

These results for the numerical investigation are discussed in the order given below.

6.1.1 Back Propagation – Train and Test

Figure 6.1-1 shows the effect on RMS error of changing architecture in the training phase. By examining the second layer hidden node axis it may be seen that the greater the number of second layer nodes the lower the accuracy and the greater the RMS error. From the above graph it may be seen that the optimum arrangement for this data set is 8 first layer hidden-nodes and two second-layer hidden-nodes. The resultant error for this arrangement is 0.05813.



(i) **(ii)**

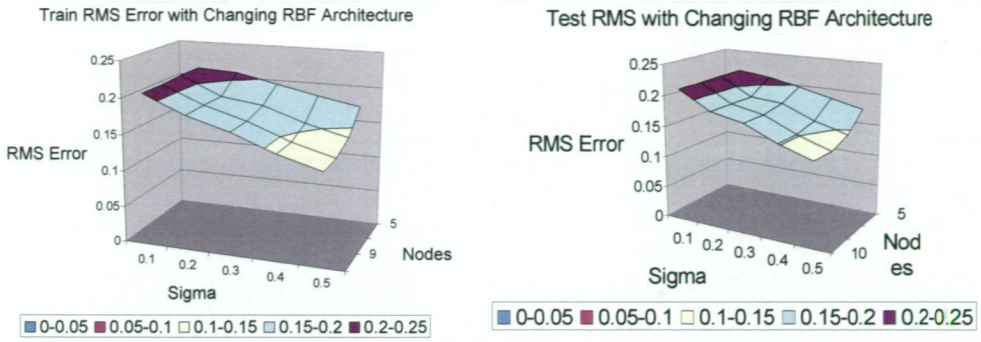
Figure 6.1-1 RMS Error with Changing Architecture for (i) Training and (ii) Testing (BP)

The general shape of the RMS error for the testing data is much the same as that for the training data. The only real difference is that the RMS testing error is marginally lower in some instances for the testing case than for the training case. Once again the minimum error is found at 8 first-layer hidden-nodes and two second-layer hidden-nodes. The value of this error is 0.05956.

From these results the optimum architecture is at 8 first-layer hidden-nodes and two second-layer hidden-nodes for the Back Propagation model. This Back Propagation architecture will be used for predicting v and θ_r in this chapter.

6.1.2 Radial Basis Function – Train and Test

The initial RMS error on the RBF graph for variations of architecture are of much the same magnitude as those of the BP network. However, the drop in error due to the addition of extra hidden nodes is markedly reduced in comparison. The range for plotting these values was held below 10 nodes and sigma values of 0.5 due to instabilities in the network.



(i) **(ii)**

Figure 6.1-2 RMS Error with Changing Architecture for Train and Test (RBF)

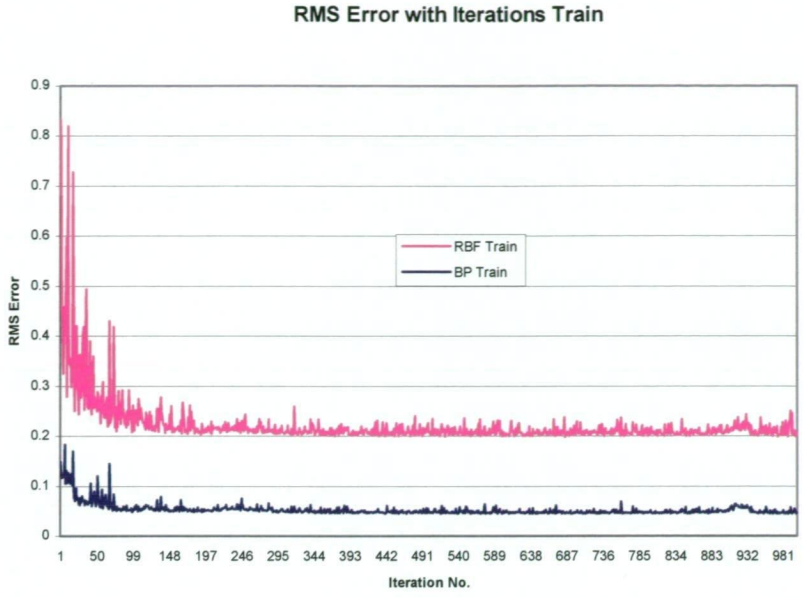
Figure 6.1-2 shows that the lowest RMS error was found to be for 10 nodes and a value of sigma equal to 0.5. This RMS value was 0.1275 more than double the value obtained for the BP training network.

The same trend persists with the testing data. The slope here appears more gradual but unlike the BP test results, the errors start fractionally larger than the training errors at just over 0.2 and decrease to the lowest value of 0.13429.

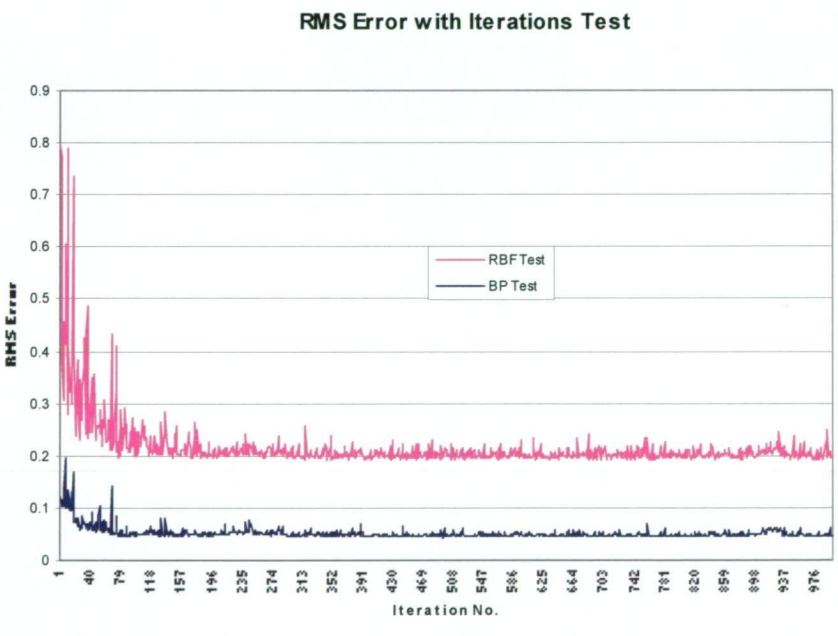
The optimum architecture is therefore selected as 10 hidden nodes and sigma equal to 0.5. This RBF architecture will be used for predictive purposes for velocity and roll angle.

6.2 Effect of iterations on RMS Error

The number of iterations completed has a large impact on the time taken to train a model but may also impact on accuracy to a degree as will be shown. The following graphs were taken from the error files of the optimum BP and optimum RBF architectures. These graphs give an indication of how a network converges to its final solution as well as an insight into the variation in RMS error magnitude that occurs even after the model has apparently converged.



(i)



(ii)

Figure 6.2-1 RMS Error with Iterations for BP and RMS (i) Train and (ii) Test

Comparison of the RMS error with iterations Figure 6.2-1 for the (ii) testing and (i) training data shows a strong correlation between the two. There are small differences in the actual values but on the whole the trends are very similar. It appears from the graphs that the major reduction in RMS error has occurred by 100 iterations, however the results seem most settled by 500 iteration. To minimise the time constraints it may

possible to limit iterations at 100. It appears that the loss of accuracy would not be as significant when considering the RMS error variation is as much as 0.02 after convergence around 850 iterations.

It may be seen from the graphs that the BP results start at around 0.18 and drop to 0.05, the RBF results start at around 0.9 and drop eventually to around 0.14. These large initial errors may be an indicator of the model instability. It is useful to note that the error values are close to their minimum by iteration 100, although 200 iterations would probably be ideal, the results from 100 iterations are likely to be reliable.

In order to best represent these velocity and roll angle values appropriate "testing course" selection is required. The following section highlights the course selection for the measurements to develop a knowledge base for neural network modelling.

6.3 Choice of Course

A number of different courses were tested in the research process. Data was gathered for each of the courses and much of it used for predicting a number of parameters such as lateral and longitudinal acceleration, yaw angle [85] and brake pressures [96]. The courses include straight-line acceleration, sweeping ovals, tight left and right-hand circles and figure 8's.

For the problem of overturning, the straight-line data could be used to estimate longitudinal velocity, however, very few vehicles are likely to overturn whilst travelling a perfectly straight line. In addition to this the variation of roll angle is minimal in this case and due mainly to engine vibration. So only one parameter could be estimated from this data set.

The tight circles and ovals had the advantage over the straight-line data that they could be used to estimate longitudinal velocity and roll angle. Each set of data represents turning in one direction only, left-hand corners or right-hand corners, so the range of data is limited in this sense.

The final set of data is the 'figure 8' test data. It has the same advantage of the circular tests being able to provide data for both longitudinal velocity and roll angle but with the added complexity of turns in both directions. This means that the network is trained and tested on a much more demanding data set than a single, continuous circular motion. This is important to highlight the relevance and utility of the results to automotive systems development. As such the figure 8 course was the final data set chosen for this analysis.

6.4 Derivation of Estimated Velocity

The ideal sensor to record longitudinal velocity is an optical cross-correlation sensor. As this was not available within the scope of this research, the velocity was estimated using a procedure that has been tested on a front wheel drive vehicle fitted with an optical cross-correlation sensor. The function used has been modified slightly to account for the test vehicle being rear wheel drive.

Based on the findings of Porcel et al.[13] the longitudinal velocity used for training the neural networks was derived using two main indicators. The velocity was taken from a single or combination of wheel speeds based on indications of the vehicle behaviour. The two indicators considered critical were loss of contact while cornering, I, and oversteering, front wheel sliding out and lateral sliding, J. While these indicators were developed for a fuzzy classification procedure, the process simply swaps the measured sensor between wheels. The system used here is based on using the best wheel speed sensors in any probability of problems as these are expected to give at least as good results as the wheel speed in question. For example, if the wheel in question has not lost contact with the road the alternate arrangement is reasonable and in the instance that it has, the results will be more accurate. The indicators mentioned are depicted in the following functions.

6.4.1 Loss of Contact While Cornering, I.

The determination of loss of contact while cornering was based on lateral acceleration L and yaw rate, Y. The visual basic function used to determine this indicator in excel is as follows:

Function I(L, Y)

```
If L = 0 And Y = 0 Then I = 0        'Normal cornering
If L > 0 And Y = 0 Then I = 1        'Loss of contact on the right side
If L < 0 And Y = 0 Then I = -1       'Loss of contact on the left side
If Y > 0 Then I = 0                  'Normal cornering
If Y < 0 Then I = 0                  'Normal cornering
End Function
```

Under normal cornering a vehicle will experience lateral acceleration and will continue to turn. Loss of contact occurs when the inside wheel of the vehicle leaves the ground. The probability of this occurring is highest when the vehicle is experiencing a lateral acceleration but not continuing to turn. The function given allows this to be indicated from the data set.

6.4.2 Oversteering, Front Wheel Sliding Out and Lateral Sliding, J.

The function to determine the indicator J, was based on steering wheel angle, a, lateral acceleration, L, the derivative of yaw rate, dd, and the derivative of lateral acceleration Ld. The function is as follows:

```
Function J(a, L, dd, Ld)
If dd = 0 And Ld = 0 Then J = 0      'Normal cornering
If dd < 0 And Ld < 0 Then J = 0      'Normal cornering
If dd > 0 And Ld > 0 Then J = 0      'Normal cornering
If L > 0 And a > 0 Then J = 0        'Normal cornering
If L < 0 And a < 0 Then J = 0        'Normal cornering
If dd < 0 And Ld = 0 Then J = -1     'Oversteering or sliding out to the right
If dd < 0 And Ld > 0 Then J = -1     'Oversteering or sliding out to the right
If L > 0 And a < 0 Then J = -1       'Oversteering or sliding out to the right
If dd > 0 And Ld = 0 Then J = 1      'Oversteering or sliding out to the left
If dd > 0 And Ld < 0 Then J = 1      'Oversteering or sliding out to the left
If L < 0 And a > 0 Then J = 1        'Oversteering or sliding out to the left
End Function
```

As shown in the function above, the probability of sliding out or oversteering can be indicated by three main parameters. These are the derivative of yaw rate, which is the rate at which a vehicle is turning, the derivative of lateral acceleration, which is the

rate at which the lateral acceleration on the vehicle is changing and steering wheel angle.

6.4.3 Velocity Function, V_{est}

The velocity function is then calculated based on the two indicators and the four wheel speeds denoted V_{12} with the first position: f indicating a front wheel, r indicating a rear wheel, and in the second position: r, and l indicating right and left respectively. I and J are the indicators as previously outlined. The values were calculated in an excel spreadsheet and referenced to allow individual calculation for each data point. A comparison of estimated velocity and average rear wheel speed is shown in Figure 6.4-1.

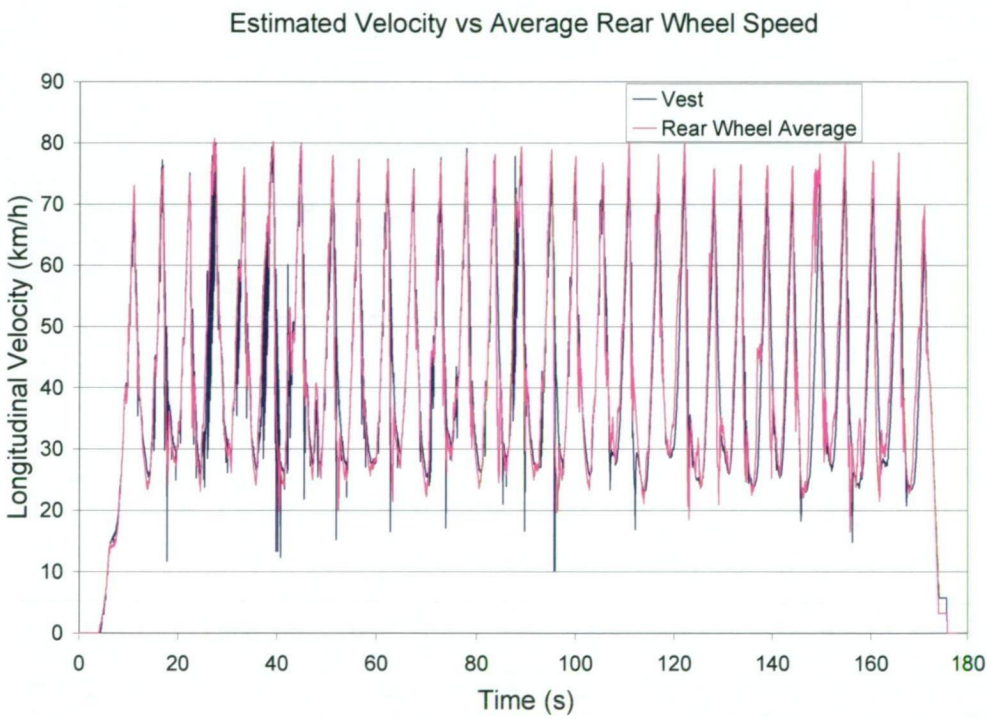


Figure 6.4-1 Comparison of Estimated Velocity and Average Rear Wheel Speed.

Figure 6.4-1 shows a comparison of the estimated velocity with the average wheel speed. V_{est} appears to be marginally noisier than the rear-wheel average; this is to be expected as the data contains information from 2 extra sensors. The estimated velocity gives a more accurate approximation of the longitudinal velocity than a simple rear wheel average as it takes more information into account. Figure 6.4-1 clearly shows the over-estimation of velocity by the average rear wheel speeds. This may be due to one of the rear wheels slipping to some degree in early acceleration.

The velocity function, V_{est} is as follows:

```
Function Vest(Vfl, Vfr, Vrl, Vrr, J, I)
If J > 0 Then Vest = (Vfr + Vrl) / 2        'Sliding or oversteering to the left
If J < 0 Then Vest = (Vfl + Vrr) / 2        'Sliding or oversteering to the right
If I > 0 Then Vest = Vrl                    'Loss of contact on the right side
If I < 0 Then Vest = Vrr                    'Loss of contact on the left side
If I = 0 Then Vest = (Vfl + Vfr) / 2        'Normal driving
End Function
```

This function allows the velocity to be taken from the most appropriate wheel speed or speeds depending on the state of the vehicle as shown by the indicator functions. The four wheel speeds were removed from the input data following the calculation of V_{est}, which was used as the output variable.

The parameters chosen to train the networks were slightly different for the two cases, velocity v, and roll angle, θ_r. Primarily it was considered important to use inputs from sensors that were different from the sensors used to find the training and test comparison outputs. In the case of θ_r this was simple as the acceleration sensor uses different processes to measure angles, acceleration and angular rates. In the case of velocity prediction, a function was used through the MoTeC software to determine the gear based on wheel speeds and engine RPM. Apart from this wheel speeds were only used in the output. The following input parameters are selected for prediction of velocity. The extent and ranges of these tested parameters are as follows:

Table 6.4-1 Parameters for Velocity Prediction

Parameter	Min	Max	Unit
Engine RPM	210	11124	rpm
Throttle Position	0	100	%
Rear Brake Hydraulic Pressure	0	4310	kPa
Front Brake Hydraulic Pressure	0	5500	kPa
Steering Wheel Angle	-192.5	209.7	deg
Suspension Position Front Left	-3.2	30.6	mm

Suspension Position Front Right	-21.6	25.9	mm
Suspension Position Rear Left	-34.1	8.7	mm
Suspension Position Rear Right	-15.5	17	mm
Longitudinal Acceleration	-2.88	1.76	G
Lateral Acceleration	-3.84	3.64	G
Vertical Acceleration	-1.48	4.68	G
Roll Angle	-0.5	9.4	deg
Pitch Angle	-2.8	130.3	deg
Yaw Angle	-71.8	537.6	deg
Gear	0	4	
Longitudinal Velocity Estimate, V_{est}	0	80.75	km/h

The number of data patterns used was 3461 for training both models. The test data was 5% of this data randomly selected and withheld from the training process. The neural network model parameters are given in Table 6.4-2.

Table 6.4-2 Neural Network Model Parameters

BP		RBF	
Inputs	16	Inputs	16
Output	1	Outputs	1
Hidden nodes 1st layer	10	Hidden nodes 1st layer	10
Transfer function	Sigmoid	Sigma	0.5
Hidden nodes 2nd layer	2		

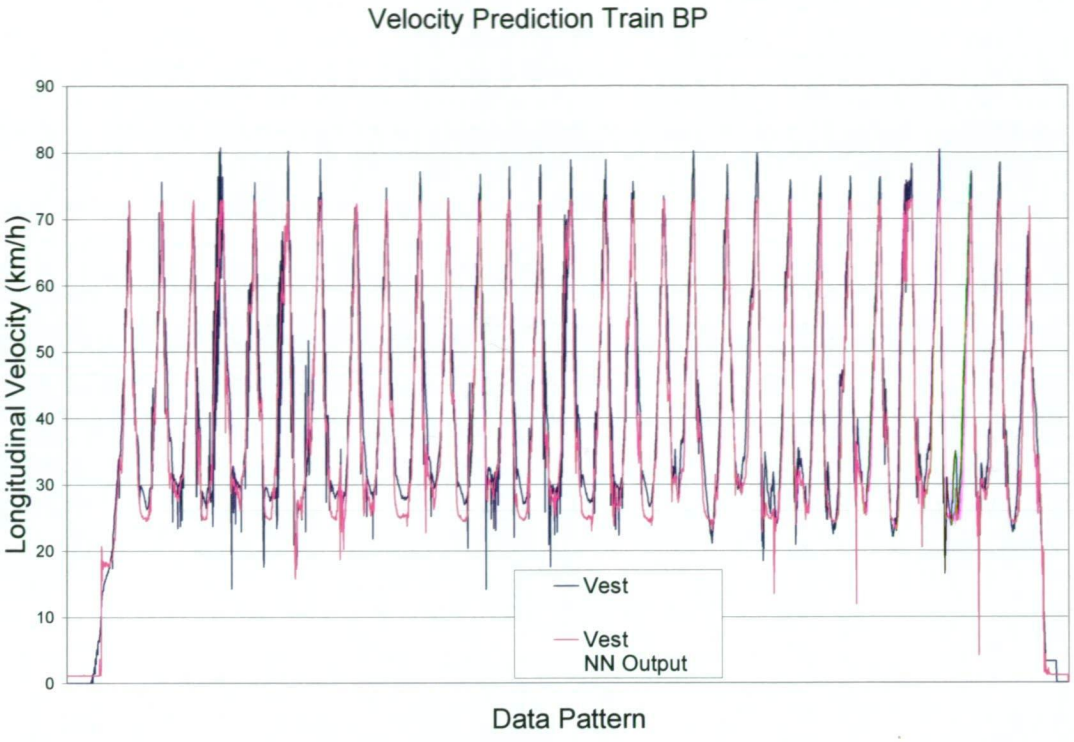
This gives a consolidated table of architecture parameters for two models. The results of testing and training were collected and collated.

6.5 Prediction of v using BP and RBF models

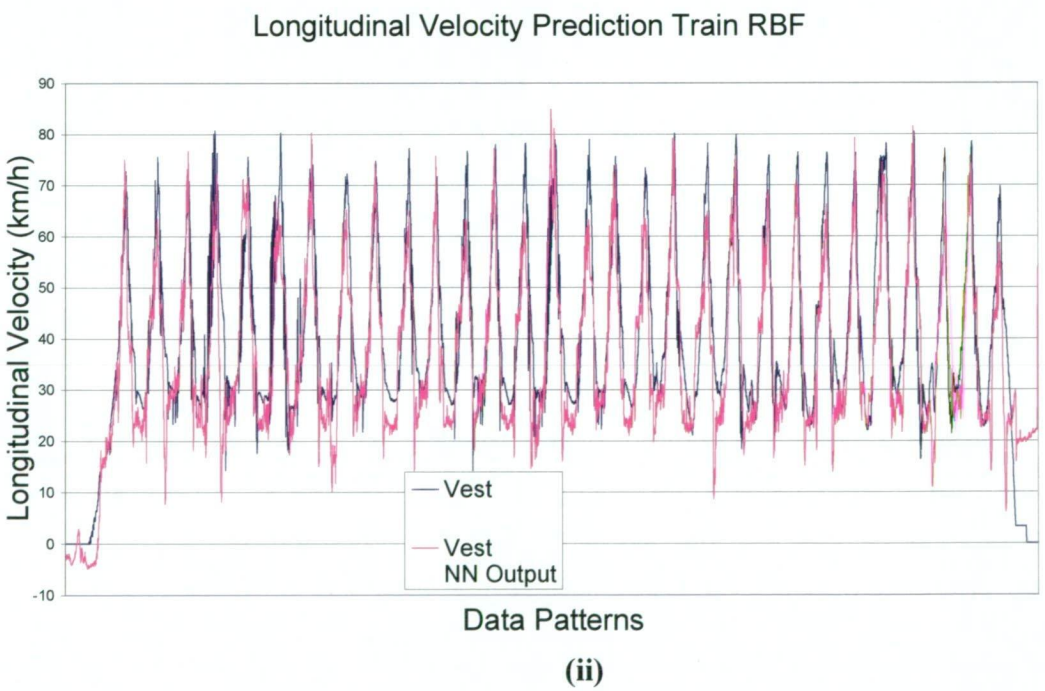
This section highlights prediction of v using both the neural network models. The training state will be shown in the first instance followed by the testing capabilities of both networks.

6.5.1 Training Results

For the training phase of velocity prediction the velocity is estimated using the logic in section 6.4.3 and the neural networks trained. The results are as follows:



(i)



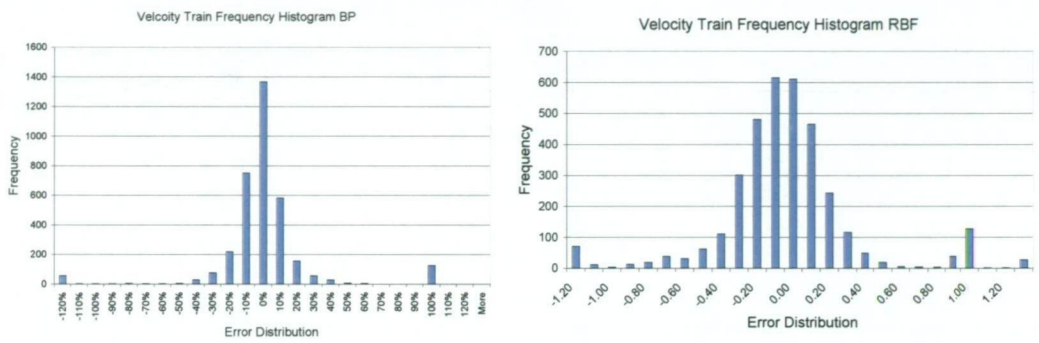
(ii)

Figure 6.5-1 Training Results using (i) BP and (ii) RBF Networks for Velocity

As shown in Figure 6.5-1 (i) training the BP network proceeded extremely well and there appears to be very little estimation of noise in the training output. This first

comparison shows that the network was able to predict the values used in the training process, an important first step.

The RBF network results also show a close estimation of the data trends. Clearly there is some discrepancy around the data extremes such as when the velocity is zero early in the data set and at the end of the data set. This appears to be an anomaly of the RBF training network. The middle data trends are well represented although the trends do not appear to be as clean as the BP training set.



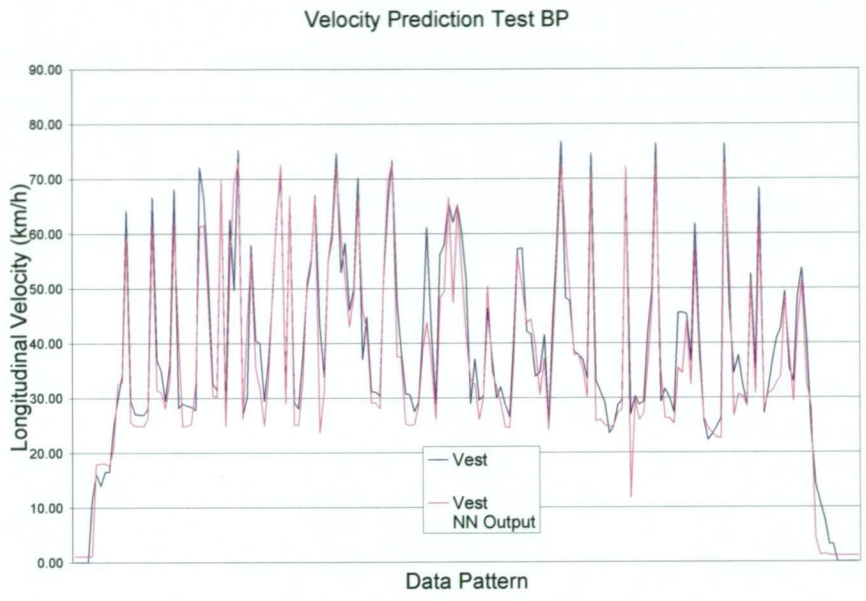
(i) (ii)

Figure 6.5-2 Error Frequency Histogram for (i) BP and (ii) RBF Velocity Train.

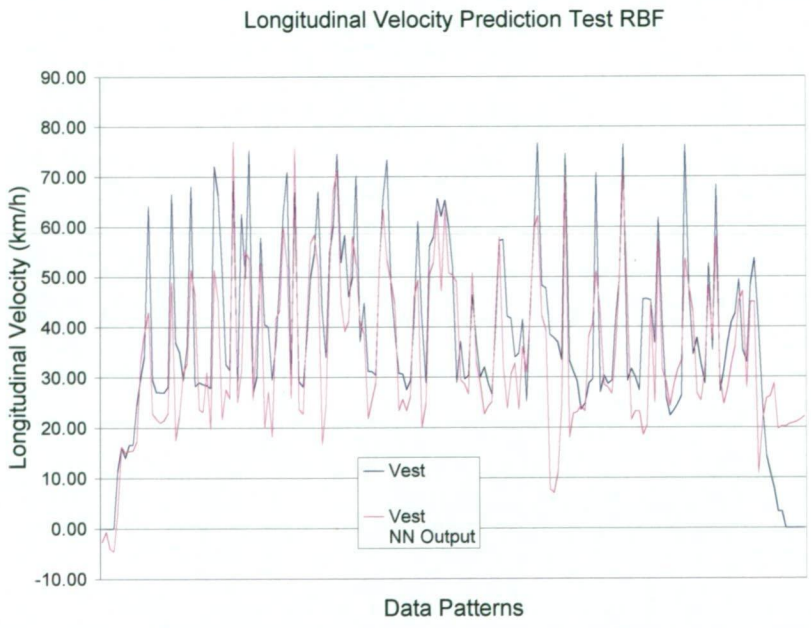
In comparing the error frequency histograms (Figure 6.5-2) the minor differences between the two models become more apparent. BP has much better predictive capability compared to RBF. Using BP, the majority of errors were well under ± 5%. This is particularly encouraging since the model is not biased for either under prediction or over prediction. This is excellent prediction considering other numerical techniques such as finite element modelling can only boast accuracy to around 15%. By contrast the RBF network appears to have the majority of errors within ± 20% with some errors distributed out to ± 40 %, Figure 6.5-2 (ii). This may be attributed to the lack of convergence of local minima solution for the sigma values chosen. This also shows the inability of the network to process such highly non-linear dynamic data.

6.5.2 Testing Results

In the testing phase, the training network is tested against a set of data tat was included in the training set. In this way the network may be tested for its ability to interpolate or generalise to new data, what it has learned from the training phase. The results obtained for the testing phase of velocity prediction are as follows:



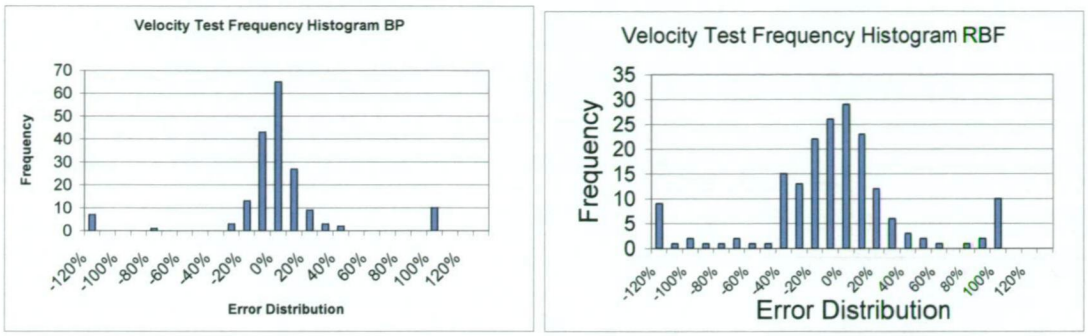
(i)



(ii)

Figure 6.5-3 Test Result using (i) BP and (ii) RBF Networks for Velocity

In Figure 6.5-3 (i) the BP network shows excellent predictive capabilities with respect to the testing data. Values in the main body of the data appear to be within 2-3 km/h of the testing data. Using BP, the majority of errors were under ± 5% Figure 6.5-4 (i). Again, this is excellent prediction.



(i) (ii)

Figure 6.5-4 Error Frequency Histogram for (i) BP and (ii) RBF Velocity Test

The variation between the RBF estimation and the original data points is more pronounced Figure 6.5-3 (ii). The RBF network appears to have the majority of errors within ± 20% with some errors distributed out to ± 40 % Figure 6.5-4 (ii). For RBF there is a variation of up to 38 km/h in the main body of the data. At the extremes the network estimation does not predict the zero velocity at all. In some applications this area may not be of concern but in an automotive application, it may be critical. In this case, the network is out by as much as 20 km/h at the extreme, a significant and possibly dangerous amount in this type of application.

The Back Propagation network is found to be superior compared to RBF for predicting v. It is important to note that this estimation of v can be integrated into a "roll-over" warning system, after substantial knowledge base is developed. The next section will deal with prediction of θ_r.

6.6 Prediction of Roll Angle Using BP and RBF Models

The roll angle of the vehicle is measured using the angle sensor. Hence the network may be trained using these values directly. The input parameters used in training the network to predict θ_r are as follows:

Table 6.6-1 Extent of Experimentation for θ_r

Parameter	Max	Min	Unit
Engine RPM	10860	2484	rpm
Throttle Position	100	0	%
Steering Wheel Angle	210.5	-184	deg
Brake Hydraulic Pressure Front	4690	0	kPa
Brake Hydraulic Pressure Rear	3869	0	kPa
Suspension Position Front Left	6000	-9.8	mm
Suspension Position Front Right	16.7	-10.4	mm
Suspension Position Rear Right	18.5	-11.6	mm
Suspension Position Rear Left	13.5	-23.5	mm
Wheel Speed Rear Right	79.3	3.7	km/h
Wheel Speed Rear Left	91.4	4.6	km/h
Wheel Speed Front Right	74.7	9.4	km/h
Wheel Speed Front Left	74.8	13.4	km/h
Lateral Acceleration	4.44	-2.25	G
Roll Angle	8.6	-3	deg

In all there are 14 inputs. These were chosen as the parameters to demonstrate the important aspects of vehicle behaviour. The complete scope of experimentation covered 6000 data patterns 5% of which were randomly selected and used as test data. The 2 layer back propagation model was trained with the specifications given in Table 6.6-2.

Table 6.6-2 Neural Network Model Parameters

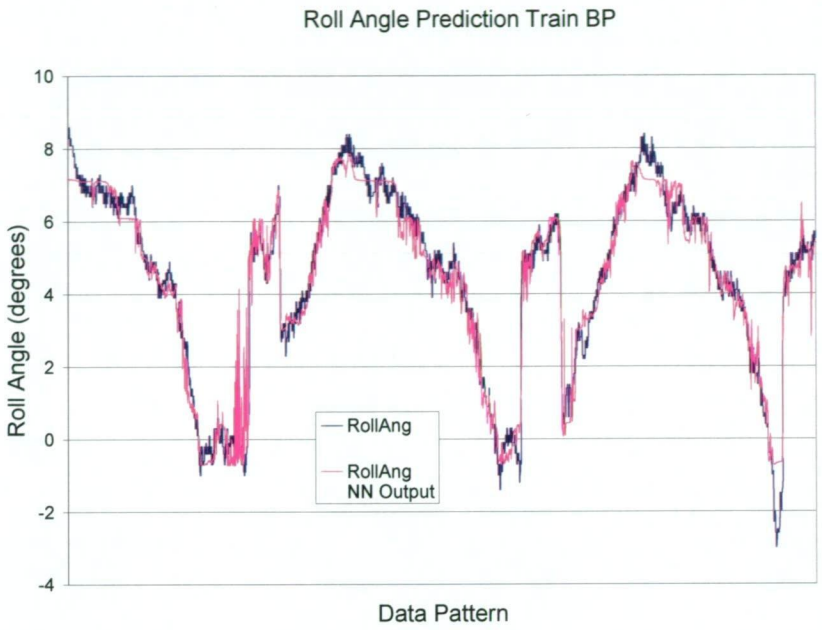
BP		RBF	
Inputs	14	Inputs	14
Output	1	Outputs	1
Hidden nodes 1st layer	10	Hidden nodes 1st layer	10
Transfer function	Sigmoid	Sigma	0.5
Hidden nodes 2nd layer	2		

The results of training and testing the networks based on these parameters were collected and collated. This section highlights predictions of θ_r using both the neural

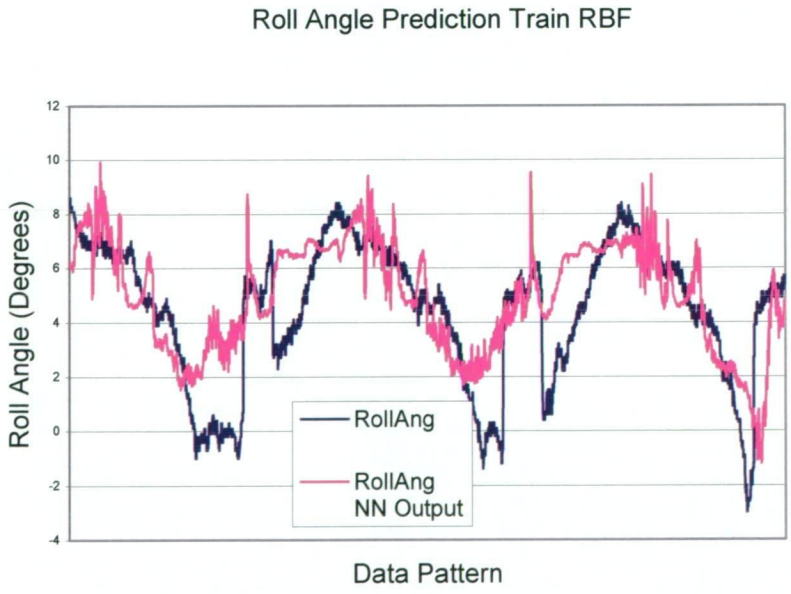
network models. The training state will be shown in the first instance followed by the testing capabilities of both networks.

6.6.1 Training Results

The training results for prediction of θ_r are as follow:



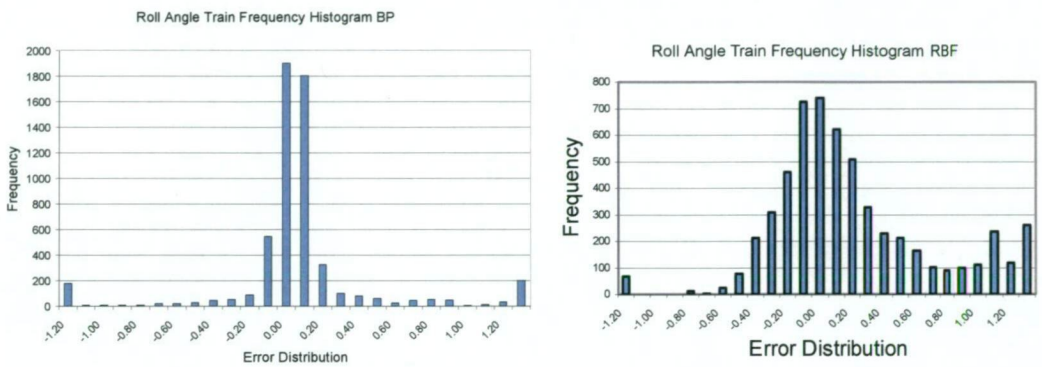
(i)



(ii)

Figure 6.6-1 Training Result using (i) BP and (ii) RBF Networks for Roll Angle

Figure 6.6-1 (i) shows the graph of training data with the BP network output in the training phase. The results are comparable for the prediction of velocity. The input data in this case appears to be much more noisy and with a much less regular trend than the velocity data. It appears that at one point the low roll angle is not well predicted. The uneven slant of the training data around zero may have contributed to this phenomenon with the normalising process minimising this effect. The difference in angle between the network result and the actual training data is a high as 5 degrees at some points. However, on the whole the variation in the main body of the data would appear to be more of the order of 0.5-1.0 degrees.



(i) **(ii)**

Figure 6.6-2 Error Frequency Histogram for (i) BP and (ii) RBF Roll Angle Train

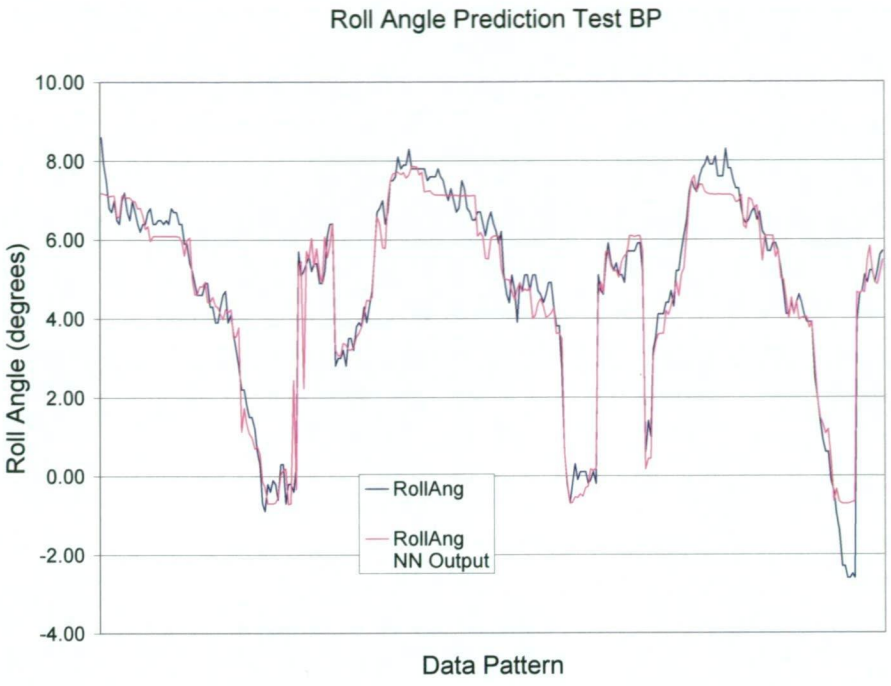
The BP histogram Figure 6.6-2 (i) shows a tight distribution largely within ± 5 %. This is very encouraging considering θ_r is an important parameter to contribute towards vehicle roll over.

The results for the RBF network Figure 6.6-1 (ii) are quite inferior in this instance. The trend comparison shows that the network does not predict θ_r well in the training case. While the absolute magnitude of errors is perhaps not larger than for the BP network, they are much more regular.

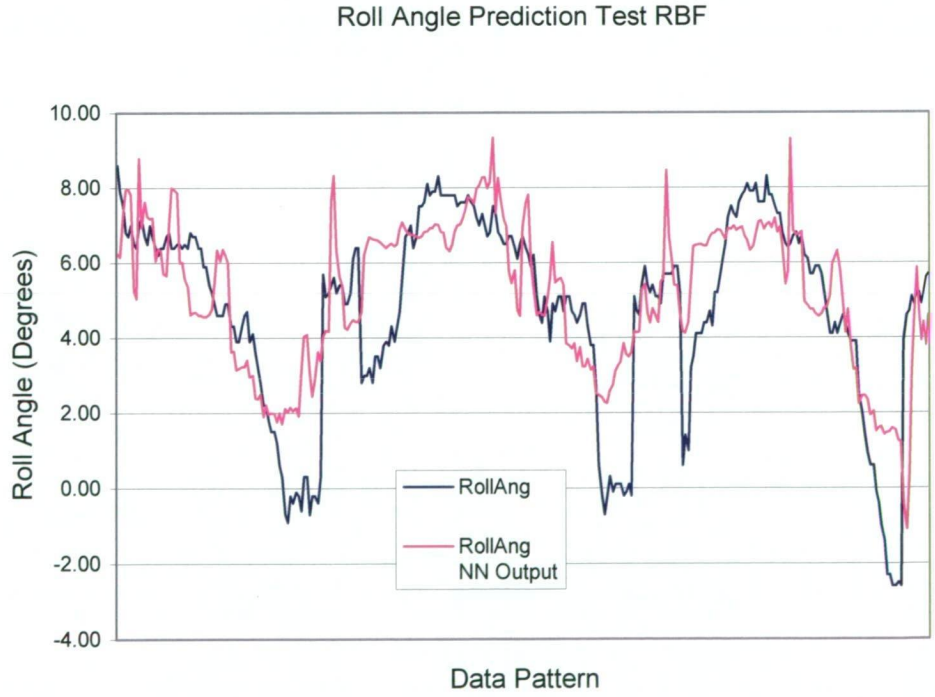
Comparison of the predictive capability as indicated by the histogram Figure 6.6-2 (ii) show that the RBF network results are basically unreliable. The errors appear to be well spread out past 40%. The model also appears to frequently over predict as shown by the high number of positive errors.

6.6.2 Testing Results

The testing results for prediction of θ_r are as follows:



(i)

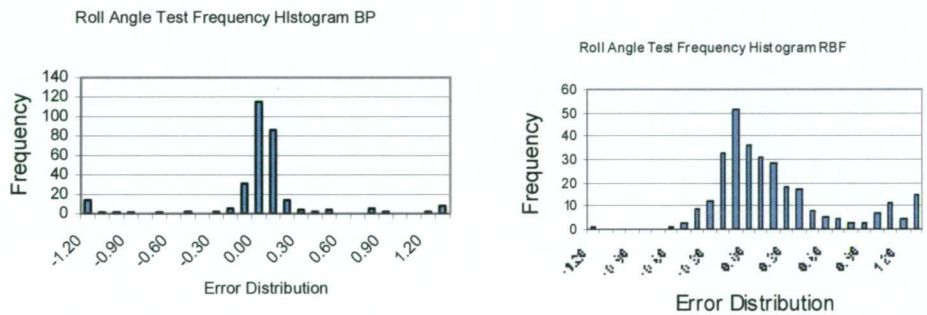


(ii)

Figure 6.6-3 Test Result Using (i) BP and (ii) RBF Networks for Roll Angle

The test results given in Figure 6.6-3 (i) echo those of the training data. There is a still an apparently instantaneous variation of 3-5 degrees in some places, but on the whole, the basic trend is very clear. Again, there appears to be an issue with the true minimum, as estimated by the BP network, being well above the actual testing data, possibly for the aforementioned reasons. In this situation 3 degrees is not significant for roll over avoidance, and best represents a practical scenario to control over turning.

It would appear that the results of the training phase for the RBF network Figure 6.6-3 (ii) are well below the standard set by the BP network. Once again, there is the underestimation of the absolute minimum, which gives a difference of around 3 degrees in this instance. The overall trend appears to be only roughly estimated with the majority of low values over predicted.



(i) **(ii)**

Figure 6.6-4 Error Frequency Histogram for (i) BP and (ii) RBF Roll Angle Test

These general comments are reinforced by the error frequency histogram. The BP network histogram Figure 6.6-4 (i) is clearly within acceptable bounds of accuracy with the majority of errors lying within ± 5%. The RBF network Figure 6.6-4 (ii) has a solid distribution of errors up to and beyond 40%. The results also show a tendency to over predict. As a predictive tool this is not acceptably accurate. In comparison with the BP network the RBF network appears quite inferior in this instance.

In the case of θ_r prediction, the BP network has been found to be superior in both training and testing phases. In comparing the two main networks, Back Propagation and Radial Basis Function, there is a variation of degree of accuracy within each network.

6.7 Concluding Remarks

Two neural networks were trained to predict values for roll angle and longitudinal velocity. The optimum architectures of those considered were found to be:

1. 8 first-layer hidden-nodes, 2 second-layer hidden-nodes, Back Propagation network with sigmoid activation function.
2. Radial Basis Function with 10 hidden nodes and sigma equal to 0.5

Table 6.7-1 Accuracy Summary

Accuracy	Velocity	Roll angle, θ_r
BP	± 5%	± 5%
RBF	± 20%	± 40%

Of the two networks tested the back propagation network gave superior results based on the results of a comparison of the trends and error frequency histograms. The importance of this preliminary investigation is to show the capability of the neural network approach to predicting velocity and roll angle. This has been achieved.

Chapter 7 Final Conclusions and Future Work

In the introduction the need for sensor control for automobiles was introduced along with the concept of the intelligent car for traffic control, navigation and prevention of vehicle roll over. Based on this, the need for reliable estimation of vehicle dynamic performance was identified and brief comment made on the current state of sensor control. The distinction between local and comprehensive control was drawn using the examples of cruise control ABS and ASS as local control strategies and the need for comprehensive control to avoid vehicle roll over highlighted. It is concluded for there is a need for intelligent tools for better control in automotive applications.

Conceptual vehicular physics were discussed as the basis for understanding the physical system. The complex nature of vehicle dynamics was highlighted along with the difficulty associated with estimation of parameters responsible for vehicle roll over. An extensive literature survey highlighted the application of neural networks and fuzzy logics to this problem and the work of Porcel et al.[13] examined in some detail as a strong foundation for the results detailed in later chapters. A general summary was made of research in artificial intelligence as used in other automotive applications such as engine control. A comparison was made of a neural network solution and a fuzzy logic solution to the question of backing a truck into a docking bay. The fuzzy rule base was found to be superior in this simple case, with the real advantage of neural networks over the fuzzy rule base being found in more complex applications. A broad overview of neural network applications in a variety of industries was also included.

The mathematical basis of the neural network technology has been discussed and the biological basis for artificial neural networks is examined. This was followed by a brief history of the development of modern neural networks and the simple two-layer Perceptron with associated functions. The different types of networks were discussed as well as some important characteristics that make the neural network approach unique compared to the conventional methods. Some considerations for improving network performance were outlined and the details of two main models discussed namely Back Propagation and Radial Basis Function. Although the models were

slightly modified to gain better momentum, the original code and algorithms were not changed.

A detailed experimental set up covering the concept, design and manufacture of the test vehicle for collection of training data, the intelligent race-car is developed. The vehicle was designed with a lower centre of gravity as the preferred option to avoid physical overturning, while the parameters that contribute to overturning were measured using sensory technology. A brief overview of the development of the frame, suspension, wheel assembly, cockpit and drive train were also discussed. More details on the development of the engine and electrical systems were covered. Safety and reliability were key aspects of the design as well as educational training of final year students.

The discussion of sensors and sensor fusion considered sensor positioning. The key sensor was shown to be the acceleration and angle sensor that was mounted as close to the vehicle centre of gravity as possible. The broad list of parameters to be measured was identified and the various sensors described and specified, beginning with the engine sensors and then the dynamic sensors. Installation and calibration of the sensors was also included. Consequently the data acquisition system was specified including telemetry, real-time clocks and software for on-line parameter estimation.

Finally, from first principles, a rationale was given for the choice of longitudinal velocity and vehicle roll angle as major parameters contributing to vehicle roll over. The optimal architecture, with numerical investigation, for back propagation was found to be 8 first-layer hidden-nodes and 2 second-layer hidden-nodes. For Radial Basis Function model the optimal architecture was found to be for 10 nodes and sigma equal to 0.5. The prediction of velocity from rear wheel speeds was based on the published work in the literature. These values were then used to train both networks. The results showed Back Propagation was the superior network for the prediction of longitudinal velocity with an RMS error distribution within $\pm$ 5 % of the required value for both training and testing, compared with the Radial Basis Function network that was distributed to $\pm$ 20% for both training and testing. Prediction of roll angle was based on the value from the angle sensor. Again both networks were

trained. The results showed Back Propagation was the superior network for the prediction of roll angle with an RMS error distribution within ± 5 % of the required value for both testing and training, compared with the Radial Basis Function network that was distributed to ± 40% for both testing and training. The importance of this preliminary investigation was to show the capability of the neural network approach to predicting longitudinal velocity and roll angle as parameters responsible for vehicle roll over. This work establishes applications of neural networks for prediction of parameters responsible for overturning.

Future work in this area will foster new avenues for control and investigation in intelligent traction control and intelligent brake pressure control. There is extensive work still to be done in development of vehicle hardware; actuators and control systems based on this technology. As a generic application, this work has established the use of neural networks as a predictive tool for estimating vehicle roll over parameters. It is both satisfying and reassuring to have a computational predictive basis for comprehensive safety systems. This work is clearly preliminary in nature, which leads to the identification and application of fast converging algorithms and other neural network models that are being continually developed in this research group.

From an industry point of view this technology would be at its most powerful when integrated with manufacturer's already available ESP and ABS control systems. The development of safer driving conditions is advantageous to researcher, manufacturer and consumer alike. This work lays the foundation for a comprehensive control system for the prevention of vehicle roll over and consequently the prevention of possible injuries and fatalities.

Chapter 8 Bibliography

[1] *Motor-vehicle safety: A 20th century public health achievement* (1999) Morbidity and Mortality Weekly Report, vol 48, issue 18, pp369-374, Pro-Quest.

[2] Lypen, John, (2000) *Intelligent car technologies,* Motor, vol. 193, issue 1, pp54-56, Pro-Quest.

[3] Jost, Kevin, (1995) *Yaw Sensing*, Automotive Engineering, September, p61.

[4] Francher, P.S.; Bareket, Z. (1998) *Evolving model for studying driver-vehicle systems performance in longitudinal control of headway*, Transportation Research Record, issue 1631, pp13-19.

[5] Von-Altrock, Constantin, (1995) *Fuzzy logic and neurofuzzy applications explained* Prentice-Hall, Upper Saddle River, New Jersey, USA.

[6] Bauer, Horst (Ed) (1999) *Driving safety systems,* Robert Bosch Gmbh, Warrendale, PA, USA.

[7] Van Zanten, Anthony; Erhardt, Rainer; Landesfeind, Klaus; Pfaff, Georg; (1999) *Stability Control*, Chapter 17 in *Automotive Electronics Handbook*, 2nd Edition, McGraw-Hill, Sydney, Australia.

[8] Wolfgang's ML Page: Vehicle Dynamics Control from ABS to ESP (1998), http://www.whnet.com/4x4/abs.html

[9] McLellan, David R.;Ryan, Joseph P.; Browalski, Edmund S. and Heinricy, John W. (1999) *Increasing the Safe Driving Envelope – ABS, Traction Control and Beyond.* Electronic Braking, Traction, and Stability Control, Automotive Electronics Serries, SAE Inc, Warrendale, PA, USA.

[10] Strickland, Alan and Dagg, Ken (1999) *ABS Braking Performance and Steering Input*, Electronic Braking, Traction, and Stability Control, Automotive Electronics Serries, SAE Inc, Warrendale, PA, USA.

[11] Mathues, Thomas (1999)*ABS Extending the Range*, Electronic Braking, Traction, and Stability Control, Automotive Electronics Serries, SAE Inc, Warrendale, PA, USA.

[12] Harrison, Graeme; (2002) *Dangers of ABS Brakes*, from web page: http://www.westprint.com.au/Travellers%20Info/vehicle_advice.htm

[13] Porcel, A.; Runde, C.; Basset, M. and Gissinger, G.L.(1998) *Neuro-fuzzy approach to real time total velocity vector estimation of a passenger car covering critical situations*, IFAC Advances in Automotive control, pp29-36.

[14] Lu, Yi; Chen, Tie Qi. and Hamilton, Brennan (1998) *A Fuzzy diagnostic model and its application in automotive engineering diagnosis*, Proceeding of the 1997 IEEE international conference on control, pp229-234.

[15] Atkinson, Chris M.; Long, Theresa W. and Hanzevack, Emil L.(1998) *Virtual Sensing: a Neural Network-Based Intelligent Performance and Emission Prediction System for On-Board diagnostics and Engine Control*, SAE Technical Paper, International Congress and Exposition, Detroit, Michigan, Feb 23-26 pp39-51.

[16] Holzmann, H; Halfmann, C.H. and Isermann, R. (1997) *Representation of 3-D Mappings for Automotive control using neural Networks and Fuzzy Logic*, Proceedings of the 1997 IEEE international conference on control applications, pp 229-234.

[17] Li, Xiaoqiu and Yurkovich, Stephen (2000) *Neural Network based, discrete adaptive sliding mode control for idle speed regulation in IC engines*, Journal of dynamic systems, measurement, and control; vol 122, pp269-275.

[18] Lenz, Ulrich and Schroder, Dierk (1998) *Air-Fuel Ratio control for Direct Injecting Combustion Engines Using Neural Networks*, International Congress and Exposition, Detroit, Michigan Feb 23-26, pp117-123.

[19] Ayeb, M. Lichtenthaler, D.; Winsel, T. and Theuerkauf, H.J.(1998) *SI Engine Modelling Using Neural Networks*, SAE publications, Danvers, MA.

[20] Zeid, Ashraf and Chang, David (1989) *A modular computer models for the Design of Vehicle dynamics control systems*, Vehicle system dynamics, 18, pp 201-221.

[21] Kosko, Bart (1992)*Neural networks and fuzzy systems : a dynamical approach to machine intelligence*, Prentice-Hall, Englewood Cliffs, N.J. USA.

[22] Nguyen, D., and Widrow, B., (1989) *The Truck Backer-upper: An Example of Self-Learning in Neural Networks*, Proceedings of International Joint Conference on neural Networks(JCNN-89), vol. II, 357-363, June. *quoted in* [21]

[23] Willis, M.J., Montague, G.A. and Peel, C., (1995) *On the application of Artificial Neural networks to Process Control*, Kluwer Academic Publishers, Boston, pp 191-219. *quoted in [24]*

[24] Frost, Fred (2000) *Neural Network Applications to Aluminium Manufacturing*, Thesis Submitted for the award of PhD at the University of Tasmania.

[25] Lee, S. and Park, J., (1991) *Neural Computation for Collision Free Path Planning*, Journ. Intelligent Manufacturing, vol. 2 no. 5 , pp.15-326. *quoted in* [24]

[26] Chryssolouris, G., Domroesse, M. and Beaullieu, P., (1992) *Sensor Synthesis for Control of Manufacturing Processes*, Journ. Engineering for Industry, vol. 114, May, pp.158-174.

[27] Sorsa, T., Koivo, N. and Koivisto, H., (1991) *Neural Networks in Process Fault Diagnosis*, IEEE Trans. Syst., Man. and Cybern., vol 21, no. 4, pp815-825. *quoted in [24]*

[28] Ray, A. K., (1991), *Equipment Fault Diagnosis: A Neural Network Approach*, Computers in Industry, vol. 16, pp 167-178 *quoted in [24].*

[29] Wasserman, P. D., Unal, A. and Haddad, S., (1991), *Neural Networks for On-Line Machine Condition Monitoring*, Intelligent Engineering Systems Through Artificial Neural Networks, New York, ASME Press, pp. 693-700, *quoted in[24].*

[30] Kotani, m., Ueda, Y., Matsumoto, h. and Kanagawa, T., (1995), *Acoustic Diagnosis for Blower with Wavelet Transform and Neural Networks*, IEEE Inc. *quoted in [24].*

[31] Burke, L. I. and Rangwala, S., (1991) *Tool Condition Monitoring in Metal Cutting: A Neural Network Approach*, Journ. Intelligent Manufacturing, vol. 2, no. 5, pp.269-280, *quoted in [24]*

[32] Elanayar, S. and Shin, Y. C., (1991), *Tool Wear Estimation in Turning Operations Based on Radial Basis Functions*, Intelligent Engineering Systems through Artificial Neural Networks, New York, ASME Press, pp. 685-692, *quoted in [24].*

[33] Govekar, E. and Peklenik, H., (1989), *Monitoring of a Drilling process by Neural Network*, 21st CIRP Int. Seminar on Manuf. Systems, Stockholm Sweden, Jun. 07-10, *quoted in [24].*

[34] Karri, V., (1996), *Performance Estimation in Wood Machining Using ANN*, Proc. Advanced Manufacturing Processes, Systems and Technologies (AMPST), Bradford, England, Mar. 21- 24, pp. 271-276.

[35] Islam, M. M., Rahman, S. M. and Sarker, R., (1998) *A Neural Network Model for Invariant Pattern Recognition*, Proc. International Conference on Computational intelligence and Multimedia Applications (ICCIMA), World Scientific, pp.318-323, *quoted in [24].*

[36] Zhizhai, H. and Zhang, M., (1998) *Self-Organising Neural Network Based Discrete Optical Flow Model for Face Recognition*, Proc. International Conference on Computational Intelligence and Multimedia Applications (ICCIMA), World Scientific, pp. 372-377, *quoted in [24]*

[37] Widrow, H. and Winter, R., (1988) *Neural Nets for Adaptive filtering and Adaptive Pattern Recognition*, Computer, vol. 21, Mar. 12, pp. 25-39, *quoted in [24].*

[38] Sobajic, D. J., Lu, J. J. and Pao, Y. H., (1988) *intelligent Control for the Intelledex 605 T Robot Manipulator*, Proc. IEEE Int. Neural Networks Conf., vol. 2, pp613-640, *quoted in [24].*

[39] Albus, J., (1975) *A New Approach to Manipulator Control: The Cerebellar model Articulation Controller (CMAC)*, Journ. Dynamic Systems, Measurement and Control, Sept 11, pp. 220-227, *quoted in [24].*

[40] *Dynamic Stability Control*, (2002) BMW, Web page, http://www.bmw.co.za/Products/FIRST/Active/act-DSC.htm

[41] Memmer, Scott, (2002) *Traction Control*, Web page, http://www.edmunds.com/ownership/safety/articles/46352/article.html

[42] Tyre Pressure Monitoring system developed through EC Craft, Web page, http://www.help-forward.gr/events/prisma-micronano.pdf

[43]*Brakes*, (2002) BMW, Web Page, http://www.bmw.co.za/Products/FIRST/Active/act-Brakes.htm

[44] *Brake by Wire* (2002) Continental Teves, Web Page, http://www.conti-online.com/generator/www/de/en/continentalteves/continentalteves/themes/products/electronic_brake_systems/emb_en.html

[45] Fischler, Martin; Firschien, Oscar. (1987) *Intelligence, the Eye, the Brain and the Computer*, Addison Wesley Publishing Company, *quoted in* [Dobra, Mick (1996) *Artificial Intelligence will Evolve*, Web page http://www.teklearning.com/ai_evo.htm]

[46] Biological neuron, synapse Diagram (1996) Web page: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol2/cs11/article2.html

[47] Caudill, Maureen; Butler, Charles (1990) *Naturally Intelligent Systems* , A Bradford book Paramus NJ, USA

[48] McCulloch, W.S.; Pitts, W., *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics, 5, 115-133, 1943

[49] Hebb, D.O. (1949) *The Organisation of Behaviour, A Neuro Psychological Theory* John Wiley, New York, USA

[50] Rosenblatt, F. (1958) *The Perceptron: A probabilistic model for information storage and organisation in the brain* Psychological Review 65, pp386-408

[51] Widrow, B. and Hoff, M. E. (1960) *Adaptive switching circuits* Western electric show and Convention Record, part 4:96-104

[52] Nilsson, N. J. (1965) *Learning Machines: Foundations of Trainable Pattern Classifiers,* McGraw Hill, New York

[53] Amari, S.I. (1972) *Learning patterns and pattern sequences by self-organising nets of threshold elements.* IEEE Trans. Computers, C-21:1197-1206

[54] Amari S.I. (1977) *Neural theory of association and concept formation.* Biological Cybernetics, 26:175-185

[55] Fukushima, K. and Miyaka, S. (1980) *Neocognition: A self-organising neural network model for a mechanism of patter recognition unaffected by shift in position.* Biological Cybernetics, 36 (4):193-202

[56] Kohonen, T. (1977) *Associative Memory: A System-Theoretical approach.* Springer-Verlag

[57] Kohonen, T. (1984) *Self-Organisation and Associative Memory.* Springer-Verlag.

[58] Anderson, James A.; Silverstein, J.W.; Rite, S.A.; and Jones, R.S. (1977) *Distinctive features, categorical perception, and probability learning: Some applications of a neural model.* Psychological Review, 84:413-451

[59] Grossberg, S. (1977) *Classical and Instrumental Learning by Neural Networks,* volume 3, pages 51-141. Academic Press

[60] Grossberg, s. (1982) *Studies of Mind and Brain: Neural Principles of Learning perception, Development, Cognition, and Motor Control.* Reidel Press

[61] Hopfield, J.J. (1982) *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Science. 79:2554-2558

[62] Hopfield, J.J. (1984) *Neurones with graded response have collective computational properties like those of two state neurones.* Proceedings of the National Academy of Science. 81:3088-3092

[63] McClelland T.L. and Rumerhart, D.E. (1986) *Parallel distributed Processing.* MIT Press and the PDP Research Group.

[64] Neuron2 diagram of basic neural network, diagram (2001) Web page: http://www.dacs.dtic.mil/techs/neural/neural2.html

[65] Red and blue step functions, diagram (2001) Web page: http://scitec.uwichill.edu.bb/cmp/online/p21h/Lecture4/lect4.htm

[66] Kosko, Bart (1992) *Neural networks and fuzzy systems : a dynamical systems approach to machine intelligence,* Prentice-Hall, Englewood Ciffs, N.J. USA

[67] Minsky, Marvin; Papert, Seymour (1969) *Perceptrons: and introduction to computational geometry* MIT Press Cambridge Mass. USA

[68] Perceptron, Diagram (2001) Web page:
http://ei.cs.vt.edu/~history/Perceptrons.Estebon.html

[69] Dayhoff, Judith E. (1990) *Neural network architectures : an introduction,* Van Nostrand Reinhold, New York, N.Y.

[70] Huang, S.H. and Zhang, H.C. (1994) *Artificial neural networks in manufacturing : Concepts, applications, and perspectives.* IEEE Transactions on Components, Packaging and Manufacturing Technology – Part A, 17(2) pp 212-228

[71] Conceptual Comparison of Feed Forward and Recurrent Neural Networks (2001), Web page now defunct.

[72] Masters, T., (1993) *Practical Neural Network Recipes in C++*, Academic Press Inc. *Quoted in* [24]

[73] Rojas, Raul (1996) *Neural networks: a systematic introduction* Springer-Verlag, New York.

[74] Lippmann, Richard P. (1987) *An Introduction to Computing with Neural Nets* IEEE ASSP Magazine April, pp 4-22

[75] Chen, C.H., (1996) *Fuzzy Logic and Neural Network Handbook*, McGraw-Hill companies, Inc., New York

[76] Bashkirov, O.A.; Braverman, E.M.; and Muchnik, I.B.; (1964) *Potential Function Algorithms for Pattern Recognition learning Machines*, Automation and Remote Control, vol. 25, pp.692-695.

[77] Broomhead, D.S. and Lowe, D. (1988) *Multivariable Functional Interpolation and Adaptive Networks,* Complex systems, vol. 2, pp 321-355.

[78] Moody, J. and Darken, C. (1989) *Fast Learning in Networks of Locally Tuned Processing Units*, Neural Computation, vol. 1, pp 281-294.

[79] Poggio, T. and Girosi, F. (1990) *Networks for Approximation and Learning*, Proc, IEEE, vol. 78, no.9, pp. 1481-1497.

[80] Renals, S., (1989) *Radial Basis Function Network for Speech Pattern Classification*, Electronics Letters 25, pp 437-439.

[81] Wasserman, P.D., (1993)*Advanced Methods in Neural Computing*, Van Nostrand Reinhold

[82] Graphical representation of Gaussian function (2001) Web page: http://www.cim.mcgill.ca/~wsun/cg/report/node9.html

[83] Karri, V, Bullen, F, and Rossmanek, P, (2001) *Design and Construction of Formula SAE Race Car as Project Based Learning*, Proceedings of the 12th Australasian Conference on Engineering Education, 7th Australasian Women in Engineering Forum: Towards Excellence in Engineering Education Conference, Brisbane Australis, 427-432.

[84] *Formula SAE 2001 Guidelines* (2001), Society of Automotive Engineers, Inc, USA.

[85] Butler, D (2002) *Traction Control on a Race Car Using Neural Networks*, Thesis Submitted for the award of Master of Engineering Science at the University of Tasmania.

[86] *Strand 7 – Finite Element Analysis System*,(1999), Software, G+D Computing.

[87] *CadKey* (1998), Software, Baystate Technologies Inc.

[88] Butler D.A., Karri, V, (2001) *Design Analysis Between On and Off-Centre Bearing for Racecar Wheel, Using Finite Element Method*, Proceedings of the 4th Quality in Research Seminar, Depok, Indonesia, 1-8.

[89] Neben, Robert, (2001) *Sensor Fusion on a Race Car*, Thesis submitted for Diplom-Ingenieur (FH) Maschinenbau at Fachhochschule Stralsund, in collaboration with the University of Tasmania, December 2001.

[90] Jones, Nicholas (2001), Thesis submitted for Honours in Engineering (Mechanical) at the University of Tasmania, December 2001.

[91] Stockwell, Walter, (1999) *Measure a vehicle's dynamic motion*, Test and Measurement World, vol. 19, issue 3, pp. 37-40.

[92] *Throttle Position Sensor* (2002), Web Page, http://www.nettally.com/silly34/tps.htm

[93] *RS Components catalogue sheet* (2002), Produced by RS Components Pty Ltd.

[94] *The OXYTEC Zirconia Measuring Principle with Gastight Seal* (2002), Web page: http://www.enotec.com/english/HTML/zirconia_design_principles.htm

[95] *Sensors overview and systems solutions*,(2002) Micronas Web page: http://www.micronas.com/products/overview/sensors/index.php

[96] Heron, Garth (2002) *Estimation of Brake Force in an Open Wheel Racing Car Using Neural Networks*, Thesis submitted for Master of Engineering Science at the University of Tasmania.

Appendix A
Neural Network Software Source Code

A-1 Back Propagation Source Code

```
program Back_Prop_2_Hidden; {2 hidden
layers, multiple inputs/outputs}
uses Dos, crt;
const


{************************************
**********************************}

{Neural Network Parameter
Specification}


  MaxNumInputs       = 40;
  MaxNumOutputs      = 10;

  MaxTrainPatterns   = 15000;
  MaxTestPatterns    = 15000;
  MaxInputPatterns   = 2500;

  MaxMaxIterations   = 1000000;

  MaxNumHiddenNodes = 100;
  MaxNumHidden2Nodes    = 100;

{************************************
**********************************}


{Delta Rule constants}

  zeta           = 0.9;  {Controls the
learning rate, 0 < zeta < 1}
  decrate        = 0.99; {Rate of
decrease of zeta over iterations range,
decrate < 1}


{Miscellaneous constants}

  dataSeed       = 1; {Seed for Random
Number Generation for initialising
network weights}
  calcSeed       = 1; {Seed for Random
Number Generation for random selection
of training data patterns}


type

{Miscellaneous ranges}

  DataRange       =
1..MaxTrainPatterns; {Specifies range
of training data patterns}
  TestDataRange   = 1..MaxTestPatterns;
{Specifies range of test data patterns}
  InputDataRange  =
1..MaxInputPatterns; {Specifies range
of validation data patterns}
  IterationsRange =
0..MaxMaxIterations; {Specifies range
used for controlling maximum epochs}

{Network layer ranges}

  InputRange    = 0..MaxNumInputs;
{index i always used for this range}
  HiddenRange   = 1..MaxNumHiddenNodes;
{index j always used for this range}
  Hidden2Range = 1..MaxNumHidden2Nodes;
{index b always used for this range}
  OutputRange   = 1..MaxNumOutputs;
{index k always used for this range}

{Input data types}

  InputLayerType     = array[InputRange]
of real;

  OutputOutType      =
array[OutputRange] of real;

  InputPEType        = record
                         x:
InputLayerType;
                       out:
OutputOutType;
                       end;

  InPEPtr            = ^InputPEType;
                       {Structure too
large for stack, put it on the heap}

  DataType           = array[DataRange]
of InPEPtr;

  TestDataType       =
array[TestDataRange] of InPEPtr;

  InputDataType      =
array[inputDataRange] of InPEPtr;

{Hidden layer 1 data types}

  HiddenWeightType   =
array[HiddenRange, InputRange] of real;

  HiddenOutType      =
array[HiddenRange] of real;

  HiddenPEType       = record
                         w:
HiddenWeightType;
                       z: HiddenOutType;
                       end;

{Hidden layer 2 data types}

  Hidden2WeightType =
array[Hidden2Range, HiddenRange] of
real;

  Hidden2OutType     =
array[Hidden2Range] of real;

  Hidden2PEType      = record
                         r:
Hidden2WeightType;
                       s:
Hidden2OutType;
                       end;

{Output data types}

  OutputWeightType   =
array[OutputRange, Hidden2Range] of
real;

  OutputPEType       = record
                         u:
OutputWeightType;
                       y: OutputOutType;
                       end;
```

```
  DeltaErrorType    =
array[OutputRange] of Real;

var

{Network variables}

  NumInputs       : Integer;
  NumOutputs      : Integer;

  TrainPatterns   : Integer;
  TestPatterns    : Integer;
  InputPatterns   : Integer;
  MaxIterations   : Integer;
  NumHiddenNodes  : Integer;
  NumHidden2Nodes: Integer;
  EpochSize       : Integer;

  linear    : Integer;
  sigmoidal : Integer;

  DataDirectory  :string;
  ImportAnal     :string;
  ParamIndex     :string;

  dataIn         :string;
  testDataIn     :string;
  errorOut       :string;
  trainingOut    :string;
  testOut        :string;
  weightsOut     :string;
  time           :string;
  inputDataFile  :string;
  netOutputfile  :string;
  ParamSpecFile  :string;

  inData         : DataType;
{Variable associated with training data
patterns}
  testData       : TestDataType;
{Variable associated with test data
patterns}
  inputData      : InputPEType;
{Variable associated with validation
data patterns}
  inputDataVar   : inputDataType;
{Variable associated with validation
data patterns}
  hiddenNodes    : HiddenPEType;
{Variable used to control hidden layer
1 computation}
  hidden2Nodes   : Hidden2PEType;
{Variable used to control hidden layer
2 computation}
  outputNodes    : OutputPEType;
{Variable used to control output layer
computation}

  delta          : DeltaErrorType;
{Variable used to determine network
error}
  rmsError       : real;
{Variable used to determine network
error}

  iterations     : IterationsRange;
{Variable used to perform maximum
iterations}
  q              : Integer;
{Variable used in random selection of
training patterns}

  fOut           : Text;
{Variable for opening output files}
  fIn            : Text;
{Variable for opening input files}
  h1,m1,s1,hund1 : Word;
{Variables for calculating program
start time}
  h2,m2,s2,hund2 : Word;
{Variables for calculating program
finish time}


{************************** I/O
Functions **************************}

procedure OpenIn (var f: Text;
filename: string);

{Used to open specified files
containing information to be read into
the network}

begin {OpenIn}

  Assign(f, filename);
  Reset(f);

end;  {OpenIn}


procedure OpenOut (var f: Text;
filename: string);

{Used to open specified files to write
network output to}

begin {OpenOut}

  Assign(f, filename);
  Rewrite(f);

end;  {OpenOut}


{************************ Utility
Functions ************************}


function RandomOne: real;

{Returns a random number in the range -
1 to 1}

begin {RandomOne}

  RandomOne := Random * 2 - 1;

end;  {RandomOne}


function Random1To (topOfRange:
Integer): DataRange;

{Returns a random integer in the range
1 to topOfRange}

begin {Random1To}

  Random1To := Round(Random *
(topOfRange - 1)) + 1;

end;  {Random1To}


function LeadingZero (w : Word) :
String;

{Enables computation time to be
formatted correctly in output file}

var   s : String;
```

```
begin {LeadingZero}

  Str(w:0,s);

  if Length(s) = 1 then

    s:='0'+s;

  LeadingZero:=s;

end;  {LeadingZero}


{********************** Initialization
Functions **********************}

procedure InitParamSpec;

{Initialises the Parameter
Specification for the NN M.Alarc¢n 24-
11-00}

begin {InitParamSpec}

  ParamSpecFile := 'c:\psc.txt';
  OpenIn (fIn, ParamSpecFile);
  readln   (fIn, DataDirectory);
  readln   (fIn, NumInputs);
  readln   (fIn, NumOutputs);
  readln   (fIn, TrainPatterns);
  readln   (fIn, TestPatterns);
  readln   (fIn, InputPatterns);
  readln   (fIn, MaxIterations);
  readln   (fIn, NumHiddenNodes);
  readln   (fIn, NumHidden2Nodes);
  readln   (fIn, linear);
  readln   (fIn, sigmoidal);
  readln   (fIn, ImportAnal);
  readln   (fIn, ParamIndex);
  Close  (fIn);

  EpochSize     := TrainPatterns;
{Number of training data patterns
considered in each epoch}
end;  {InitParamSpec}


procedure InitFilenames;
{Initialises tFilenames for the NN
M.Alarc¢n 7-12-00}

begin {InitFilenames}
  dataIn        := DataDirectory +
ImportAnal + 'trn' + ParamIndex +
'.txt'; {Input file containing training
data set}
  testDataIn    := DataDirectory +
ImportAnal + 'tst' + ParamIndex +
'.txt'; {Input file containing test
data set}
  errorOut      := DataDirectory +
ImportAnal + 'err' + ParamIndex +
'.out'; {Output file for training and
test error}
  trainingOut   := DataDirectory +
ImportAnal + 'trn' + ParamIndex +
'.out'; {Output file for training
results}
  testOut       := DataDirectory +
ImportAnal + 'tst' + ParamIndex +
'.out'; {Output file for test results}
  weightsOut    := DataDirectory +
ImportAnal + 'wts' + ParamIndex +
'.out'; {Output file for weights
matrix}
  time          := DataDirectory +
ImportAnal + 'tim' + ParamIndex +
'.out'; {Output file for computation
time}

  inputDataFile := DataDirectory +
'input.txt';  {Input file containing
validation data set}
  netOutputfile := DataDirectory +
'output.out'; {Output file for
validation results}

end;  {InitFilenames}


procedure InitDataStrs (var d:
DataType; var t: TestDataType);

{Initialises the training and test data
structures on the heap}

var  m: DataRange;
     n: TestDataRange;

begin {InitDataStrs}

  for m := 1 to TrainPatterns do

    New(d[m]);

  for n := 1 to TestPatterns do

    New(t[n]);

end;  {InitDataStrs}


procedure DisposeDataStrs (var d:
DataType; var t: TestDataType);

{Disposes of the training and test data
structures}

var  m: DataRange;
     n: TestDataRange;

begin {DisposeDataStrs}

  for m := 1 to TrainPatterns do

    Dispose(d[m]);

  for n := 1 to TestPatterns do

    Dispose(t[n]);

end;  {DisposeDataStrs}


procedure InitData (var d: DataType;
var t: TestDataType);

{Reads in training and testing data
from specified files}

var  m: DataRange;
     n: TestDataRange;
     i: InputRange;
     k: OutputRange;

begin {InitData}

  writeln;
  writeln('Reading in the training
data');
  writeln;
  OpenIn(fIn, dataIn);
```

```
  for m := 1 to TrainPatterns do

   begin {m}

    for i := 1 to NumInputs do

      read(fIn, d[m]^.x[i]);

    for k := 1 to NumOutputs do

      read(fIn, d[m]^.out[k]);

    readln(fIn);

   end;  {m}

  Close(fIn);

  for m := 1 to TrainPatterns do

    d[m]^.x[0] := 1;

  writeln('Reading in the test data');
  writeln;
  writeln(MaxIterations,' iterations
will now commence to generate network
weights');
  writeln;
  OpenIn(fIn, testDataIn);

  for n := 1 to TestPatterns do

   begin {n}

    for i := 1 to NumInputs do

      read(fIn, t[n]^.x[i]);

    for k := 1 to NumOutputs do

      read(fIn, t[n]^.out[k]);

    readln(fIn);

   end;  {n}

  Close(fIn);

  for n := 1 to TestPatterns do

    t[n]^.x[0] := 1;

end;  {InitData}


procedure InitHidden1Layer (var h:
HiddenPEType);

{Initializes the hidden layer 1 weights
to random reals in the range -1 to 1}

var  i: InputRange;
     j: HiddenRange;

begin {InitHidden1Layer}

  for j := 1 to NumHiddenNodes do

    for i := 0 to NumInputs do

      h.w[j, i] := RandomOne;

end;  {InitHidden1Layer}


procedure InitHidden2Layer (var f:
Hidden2PEType);

{Initializes the hidden layer 2 weights
to random reals in the range -1 to 1}

var  j: HiddenRange;
     b: Hidden2Range;

begin {InitHidden2Layer}

  for b := 1 to NumHidden2Nodes do

    for j := 1 to NumHiddenNodes do

      f.r[b, j] := RandomOne;

end;  {InitHidden2Layer}


procedure InitOutputLayer (var o:
OutputPEType);

{Initializes the output weights to
random reals in the range -1 to 1}

var  b: Hidden2Range;
     k: outputRange;

begin {InitOutputLayer}

  for k := 1 to NumOutputs do

    for b := 1 to NumHidden2Nodes do

      o.u[k, b] := RandomOne;

end;  {InitOutputLayer}


{********************** Neural Net
Procedures **********************}

procedure NetForwardDel (var inp:
InputPEType; var h: HiddenPEType; var
f: Hidden2PEType;
var o: OutputPEType; var delta:
DeltaErrorType);

{Calculates network output for a given
input and also the error, delta}

var  net, val, sum: real;
     i: InputRange;
     j: HiddenRange;
     b: Hidden2Range;
     k: OutputRange;

begin {NetForwardDel}

{Hidden layer 1 forward}

  for j := 1 to NumHiddenNodes do

  begin {j}

    net := 0;

    for i := 0 to NumInputs do

      net := net + h.w[j, i] *
inp.x[i];

    h.z[j] := 1 / (1 + exp(-net));

  end; {j}
```

```
{Hidden layer 2 forward}
  for b := 1 to NumHidden2Nodes do
   begin {b}
    sum := 0;
    for j := 1 to NumHiddenNodes do
      sum := sum + f.r[b,j] * h.z[j];
    f.s[b] := 1 / (1 + exp(-sum));
   end;  {b}
{Output layer forward}
  for k := 1 to NumOutputs do
   begin {k}
    val := 0;
    for b := 1 to NumHidden2Nodes do
      val := val + o.u[k, b] * f.s[b];
    o.y[k] := sigmoidal * (1 / (1 +
exp(-val))) + linear * val;
    delta[k] := inp.out[k] - o.y[k];
   end;  {k}
end;  {NetForwardDel}


procedure NetForward (var inp:
InputPEType; var h: HiddenPEType; var
f: Hidden2PEType;
var o: OutputPEType; var
d1,d2:OutputOutType);

var  net, val, sum: real;
     i: InputRange;
     j: HiddenRange;
     b: Hidden2Range;
     k: OutputRange;

begin {NetForward}

{Hidden layer forward}
  for j := 1 to NumHiddenNodes do
    begin {j}
      net := 0;
      for i := 0 to NumInputs do
        net := net + h.w[j, i] *
inp.x[i];
      h.z[j] := 1 / (1 + exp(-net));
    end;  {j}
{Hidden layer 2 forward}
  for b := 1 to NumHidden2Nodes do
   begin {b}
    sum := 0;
    for j := 1 to NumHiddenNodes do
      sum := sum + f.r[b,j] * h.z[j];
    f.s[b] := 1 / (1 + exp(-sum));
   end;  {b}
{Output layer forward}
  for k := 1 to NumOutputs do
   begin {k}
    val := 0;
    for b := 1 to NumHidden2Nodes do
      val := val + o.u[k, b] * f.s[b];
    o.y[k] := sigmoidal * (1 / (1 +
exp(-val))) + linear * val;
    d1[k] := inp.out[k];
    d2[k] := o.y[k];
   end;  {k}
end;  {NetForward}


procedure NetTrain (var inp:
InputPEType; var h: HiddenPEType; var
f: Hidden2PEType;
var o: OutputPEType; newzeta : real);

{This procedure calculates an output
for any given input, compares the
predicted output
 with the given output, calculates the
associated error, delta, and updates
the network
 weights using the "delta rule"}

var  i: InputRange;
     j: HiddenRange;
     k: OutputRange;
     b: Hidden2Range;
     net, val, deltaJ, sum, deltaB,
deltaB1, val1: real;

begin {NetTrain}

{Hidden layer forward}
  for j := 1 to NumHiddenNodes do
   begin {j}
    net := 0;
    for i := 0 to NumInputs do
      net := net + h.w[j, i] *
inp.x[i];
    h.z[j] := 1 / (1 + exp(-net));
   end;  {j}
{Hidden layer 2 forward}
  for b := 1 to NumHidden2Nodes do
   begin {b}
    sum := 0;
    for j := 1 to NumHiddenNodes do
```

```
      sum := sum + f.r[b,j] * h.z[j];

    f.s[b] := 1 / (1 + exp(-sum));

   end;  {b}

{Output layer forward}

  for k := 1 to NumOutputs do

   begin {k}

    val := 0;

    for b := 1 to NumHidden2Nodes do

      val := val + o.u[k,b] * f.s[b];

    o.y[k] := sigmoidal * (1 / (1 +
exp(-val))) + linear * val;
    delta[k] := inp.out[k] - o.y[k];

   end;  {k}

{Update output layer weights}

  for k := 1 to NumOutputs do

    for b := 1 to NumHidden2Nodes do

      o.u[k, b] := o.u[k, b] + newzeta
* delta[k] * f.s[b];

{Update hidden layer 2 weights}

  for b := 1 to NumHidden2Nodes do

   begin {b}

    val := 0;

    for k := 1 to NumOutputs do

     val := val + delta[k] * o.u[k, b];

    deltaB := f.s[b] * (1 - f.s[b]) *
val;

    for j := 1 to NumHiddenNodes do

     f.r[b, j] := f.r[b, j] + newzeta *
deltaB * h.z[j];

   end;  {b}

{Update hidden layer 1 weights}

   for j := 1 to NumHiddenNodes do

    begin {j}

     sum := 0;
     val1 := 0;

     for b := 1 to numHidden2nodes do

      begin {b}

        for k := 1 to NumOutputs do

          val1 := val1 + delta[k] *
o.u[k, b];

        deltaB1 := f.s[b] * (1 -
f.s[b]) * val1;

        sum := sum + deltaB1 *
f.r[b,j];
        deltaJ := h.z[j] * (1 - h.z[j])
* sum;

        for i := 0 to NumInputs do

          h.w[j, i] := h.w[j, i] + zeta
* deltaJ * inp.x[i];

      end;  {b}

    end;  {j}

end;  {NetTrain}


{************************ Display
Procedures ************************}


procedure DisplayError (var rms: real;
inD: DataType; tD: TestDataType; h:
HiddenPEType;
f: Hidden2PEType; o: OutputPEType;
newzeta: real);

{Calculates the Root Mean Square, RMS,
error for all the training and test
inputs}
{Displayed on user screen for
monitoring purposes and also written to
the specified file}

var  trErr, testErr, val, testRms:
real;
     del: DeltaErrorType;
     m: DataRange;
     n: TestDataRange;
     k: OutputRange;

begin {DisplayError}

{Training data set calculations}

  trErr := 0;

  for m := 1 to TrainPatterns do

   begin {m}

    NetForwardDel(inD[m]^, h, f, o,
del);
    val := 0;

    for k := 1 to NumOutputs do

      val := val + del[k] * del[k];

    trErr := trErr + val;

   end;  {m}

   rms := Sqrt(trErr/TrainPatterns);
{RMS error over all training patterns}

{Test data set calculations}

  testErr := 0;

  for n := 1 to TestPatterns do

   begin {n}

    NetForwardDel(tD[n]^, h, f, o,
del);
    val := 0;
```

```
      for k := 1 to NumOutputs do

        val := val + del[k] * del[k];

      testErr := testErr + val;

    end;  {n}

    testRms :=
Sqrt(testErr/TestPatterns); {RMS error
over all test patterns}

  writeln(fOut, iterations : 3, '  ',
rms : 5 : 5, '  ', testRms : 5 : 5);

end;  {DisplayError}


procedure WriteOutTrainingData (var
inD: DataType; var h: HiddenPEType; var
f: Hidden2PEType;
var o: OutputPEType);

{Writes the predicted and actual
outputs from training to the specified
file}

var  curve1, curve2: OutputOutType;
     p: DataRange;
     k: OutputRange;

begin {WriteOutTrainingData}

  OpenOut(fOut, trainingOut);

  for p := 1 to TrainPatterns do

   begin {p}

    NetForward(inD[p]^, h, f, o,
curve1, curve2);

    for k :=1 to NumOutputs do

      write(fOut, curve1[k] : 7 : 5, '
');

    for k :=1 to NumOutputs do

      write(fOut, curve2[k] : 7 : 5, '
');

    writeln(fOut);

   end;  {p}

  Close(fOut);

end;  {WriteOutTrainingData}


procedure WriteOutTestData (var tD:
TestDataType; var h: HiddenPEType; var
f: Hidden2PEType;
var o: OutputPEType);

{Writes the predicted and actual
outputs from testing to the specified
file}

var  curve1, curve2: OutputOutType;
     q: TestDataRange;
     k: OutputRange;

begin {WriteOutTestData}

  OpenOut(fOut, testOut);

  for q := 1 to TestPatterns do

   begin {q}

    NetForward(tD[q]^, h, f, o, curve1,
curve2);

    for k := 1 to NumOutputs do

      write(fOut, curve1[k] : 7 : 5, '
');

    for k := 1 to NumOutputs do

      write(fOut, curve2[k] : 7 : 5, '
');

    writeln(fOut);

   end;  {q}

  Close(fOut);

end;  {WriteOutTestData}


procedure WriteOutWeights (var h:
HiddenPEType; var f: Hidden2PEType; var
o: OutputPEType);

{Writes the weights matrix to the
specified file}

var  i: InputRange;
     j: HiddenRange;
     b: Hidden2Range;
     k: OutputRange;

begin {WriteOutWeights}

  OpenOut(fOut, weightsOut);

  for j := 1 to NumHiddenNodes do

   begin {j}

    for i := 0 to NumInputs do

      write(fOut, h.w[j, i] : 8 : 5, '
');

    writeln(fOut);

   end;  {j}

  for b := 1 to NumHidden2Nodes do

   begin {b}

    for j := 1 to NumHiddenNodes do

     write(fOut, f.r[b,j] : 8 : 5,' ');

    writeln(fOut);

   end;  {b}

  writeln(fOut);

  for j := 1 to NumHiddenNodes do

   begin {j}

    for k := 1 to NumOutputs do
```

```
      Write(fOut, o.u[k,j] : 8 : 5, '
');

     writeln(fOut);

    end;  {j}

   Close(fOut);

end;  {WriteOutWeights}


{Procedures from here to "main program"
used for running option 2 of the
program}


procedure InitInputData (var d:
inputDataType);

{Reads validation data patterns from
specified file}

var  m: inputDataRange;
     i: InputRange;
     k: OutputRange;

begin {InitInputData}

  writeln('Reading in the input data');
  OpenIn(fIn, inputDataFile);

  for m := 1 to InputPatterns do

   begin {m}

    for i := 1 to NumInputs do

     read(fIn, d[m]^.x[i]);

    for k := 1 to NumOutputs do

     read(fIn,d[m]^.out[k]);

   end;  {m}

    for m := 1 to InputPatterns do

     d[m]^.x[0] := 1; {}

  Close(fIn);

end;  {InitInputData}

procedure ReadInWeights(var
h:HiddenPEType; var f:Hidden2PEType;
var o:OutputPEType);

{Reads weights matrix from file
produced using option 1 of the program}

var  i, j, b, k : integer;

begin {ReadInWeights}

  OpenIn(fIn,weightsOut);

  for j := 1 to NumHiddenNodes do

   begin {j}

    for i := 0 to NumInputs do

     read(fIn, h.w[j, i]) ;
     readln(fIn);

   end;  {j}

   for b := 1 to NumHidden2Nodes do

    begin {b}

     for j := 1 to NumHiddenNodes do

      read(fIn, f.r[b, j]);
      readln(fIn);

    end;  {b}

   readln(fIn);

    for j := 1 to NumHiddenNodes do

     begin {j}

      for k := 1 to NumOutputs do

       read(fIn, o.u[k,j]);
       readln(fIn);

     end;  {j}

  Close(fIn);

end;  {ReadInWeights}

procedure WriteOutNetOutputData (var
tD: inputDataType; var h: HiddenPEType;
var f:Hidden2PEType; var o:
OutputPEType);

{Calculates the output for any given
input using the weights developed
during training}

var  curve1, curve2: OutputOutType;
     q: inputDataRange;
     k:OutputRange;

begin {WriteOutTestData}

  OpenOut(fOut, netOutputFile);

  for q := 1 to InputPatterns do

   begin {q}

    NetForward(tD[q]^, h, f, o, curve1,
curve2);

    for k := 1 to NumOutputs do

     write(fOut, curve1[k] : 7 : 5, '
');

    for k := 1 to NumOutputs do

     write(fOut, curve2[k] : 7 : 5, '
');
     writeln(fOut);

   end;  {q}

  Close(fOut);

end;  {WriteOutTestData}

procedure InitInputDataStrs (var d:
inputDataType);

{Initialises the training and test data
structures on the heap}

var  m: inputDataRange;
```

```
begin {InitDataStrs}

  for m := 1 to InputPatterns do

   New(d[m]);

end;  {InitDataStrs}


{*************************** Main
Program ***************************}

var  answer: integer;
     count:    integer;
     stop:     integer;
     ni:       integer;
     buffer:   integer;
     newzeta : real;

begin {Main program}

 InitParamSpec; {Gets the Parameters
for the NN}

 clrscr;

 writeln('  FeedForward BackPropagation
Neural Network');

  answer:=1;

 if (answer = 2) then

  begin {2}

    InitinputDataStrs (inputDataVar);
    ReadInWeights (HiddenNodes,
Hidden2Nodes, OutputNodes);
    InitInputData (inputDataVar);
    WriteOutNetOutputData
(inputDataVar, HiddenNodes,
Hidden2Nodes, OutputNodes);

  end   {2}

 else

  begin {1}
   ni:=NumInputs;
   stop:=ni*2;
   for count:=0 to stop do
   begin {count}
   if count=0 then
      begin
      ImportAnal:='';
      ParamIndex:='';
      InitFilenames;
      end;
   if (count>0) and (count<=ni) then
      begin
      ImportAnal:='pi';
      str(count,ParamIndex);
      NumInputs:=ni-1;
      InitFilenames;
      end;
   if count>ni then
      begin
      ImportAnal:='ci';
      buffer:=count-ni;
      str(buffer,ParamIndex);
      NumInputs:=ni-1;
      InitFilenames;
      end;

    GetTime (h1, m1, s1, hund1);

    {Initialize the training and test
sets}

    InitDataStrs (inData, testData);
    InitData (inData, testData);
    OpenOut (fOut, errorOut);

    {Initialise all weights using
random values}

    randSeed := dataSeed;
    InitHidden1Layer (hiddenNodes);
    InitHidden2Layer (hidden2Nodes);
    InitOutputLayer (outputNodes);

    {A 'for' loop is used to fix the
total number of iterations}

    {Reset the random seed so that the
calling sequence can be contolled}

    randSeed := calcSeed;
    newzeta := zeta;

    for iterations := 1 to
MaxIterations do

     begin {iterations}

      for q := 1 to EpochSize do

       begin {q}

        inputData :=
InData[Random1To(TrainPatterns)]^;
       NetTrain (inputData,
hiddenNodes, hidden2Nodes, outputNodes,
newzeta);

       end;  {q}

      {Display RMS error for each
iteration}

      DisplayError (rmsError, inData,
testData, hiddenNodes, hidden2Nodes,
outputNodes, newzeta);

      {Decrease delta rule constant
over iterations range}

      newzeta := decrate * newzeta;

      if newzeta < 0.1 then

       newzeta := 0.1

      else

       newzeta := newzeta;

     end;  {iterations}

    Close (fOut);
    WriteOutTrainingData (inData,
hiddenNodes, hidden2Nodes,
outputNodes);
    WriteOutTestData (testData,
hiddenNodes, hidden2Nodes,
outputNodes);
    WriteOutWeights (hiddenNodes,
hidden2Nodes, outputNodes);
    DisposeDataStrs (inData, testData);
    GetTime (h2, m2, s2, hund2);
    OpenOut (fOut, time);
    writeln (fOut, 'Start time  :-
',LeadingZero(h1),':',LeadingZero(m1),'
```

```
:',LeadingZero(s1),':',LeadingZero(hund
1));
    writeln (fOut, 'Finish time :-
',LeadingZero(h2),':',LeadingZero(m2),'
:',LeadingZero(s2),':',LeadingZero(hund
2));
    Close(fOut);
    end;  {count}
  end; {1}
    Assign(fIn,'c:\psc.txt');
    Erase(fIn);

end.  {Main program}
```

A-2: Radial Basis Function Source Code

```
program Radial_Basis_Function;
{multiple inputs/outputs, Gaussian
function}
uses Dos, crt;
const


{************************************
**********************************}

{Neural Network Parameter
Specification}
  MaxNumInputs       = 40;
  MaxNumOutputs      = 10;

  MaxTrainPatterns   = 15000;
  MaxTestPatterns    = 15000;
  MaxInputPatterns   = 2500;

  MaxMaxIterations   = 1000000;

  MaxNumHiddenNodes = 100;

{************************************
**********************************}


{Delta Rule constants}

  zeta           = 0.9;  {Controls the
learning rate, 0 < zeta < 1}
  decrate        = 0.99; {Rate of
decrease of zeta over iterations range,
decrate < 1}


{Miscellaneous constants}

  dataSeed       = 1; {Seed for Random
Number Generation for initialising
network weights}
  calcSeed       = 1; {Seed for Random
Number Generation for random selection
of training data patterns}

type

{Miscellaneous ranges}

  DataRange      =
1..MaxTrainPatterns; {Specifies range
of training data patterns}
  TestDataRange   = 1..MaxTestPatterns;
{Specifies range of test data patterns}
  InputDataRange  =
1..MaxInputPatterns; {Specifies range
of validation data patterns}
  IterationsRange =
0..MaxMaxIterations; {Specifies range
used for controlling maximum epochs}

{Network layer ranges}

  InputRange  = 1..MaxNumInputs;
{index i always used for this range}
  HiddenRange = 0..MaxNumHiddenNodes;
{index j always used for this range}
  OutputRange = 1..MaxNumOutputs;
{index k always used for this range}

{Input data types}

  InputLayerType   = array[InputRange]
of real;

  OutputOutType    = array[OutputRange]
of real;

  InputPEType      = record
                       x:
InputLayerType;
                     out:
OutputOutType;
                     end;

  InPEPtr          = ^InputPEType;
                     {Structure too
large for the stack, put it on the
heap}

  DataType         = array[DataRange]
of InPEPtr;

  TestDataType     =
array[TestDataRange] of InPEPtr;

  InputDataType    =
array[inputDataRange] of InPEPtr;


{Hidden data types}

  HiddenWeightType = array[HiddenRange,
InputRange] of real;

  HiddenOutType    = array[HiddenRange]
of real;

  HiddenPEType     = record
                       w:
HiddenWeightType;
                     z: HiddenOutType;
                     end;

{Output data types}

  OutputWeightType = array[OutputRange,
HiddenRange] of real;

  OutputPEType     = record
                       u:
OutputWeightType;
                     y: OutputOutType;
                     end;

  DeltaErrorType   = array[OutputRange]
of Real;

var

{Network variables}
  NumInputs      : Integer;
  NumOutputs     : Integer;

  TrainPatterns  : Integer;
  TestPatterns   : Integer;
  InputPatterns  : Integer;
  MaxIterations  : Integer;
  NumHiddenNodes : Integer;
  EpochSize      : Integer;
  sigma : real;

  DataDirectory  :string;
  ImportAnal     :string;
  ParamIndex     :string;
```

```
  dataIn          :string;
  testDataIn      :string;
  errorOut        :string;
  trainingOut     :string;
  testOut         :string;
  weightsOut      :string;
  time            :string;
  inputDataFile   :string;
  netOutputfile   :string;
  ParamSpecFile   :string;

  inData          : DataType;
{Variable associated with training data
patterns}
  testData        : TestDataType;
{Variable associated with test data
patterns}
  inputData       : InputPEType;
{Variable associated with validation
data patterns}
  inputDataVar    : inputDataType;
{Variable associated with validation
data patterns}
  hiddenNodes     : HiddenPEType;
{Variable used to control hidden layer
computation}
  outputNodes     : OutputPEType;
{Variable used to control output layer
computation}

  delta           : DeltaErrorType;
{Variable used to determine network
error}
  rmsError        : real;
{Variable used to determine network
error}

  iterations      : IterationsRange;
{Variable used to perform maximum
iterations}
  q               : Integer;
{Variable used in random selection of
training patterns}

  fOut            : Text;
{Variable for opening output files}
  fIn             : Text;
{Variable for opening input files}

  h1,m1,s1,hund1 : Word;
{Variables for calculating program
start time}
  h2,m2,s2,hund2 : Word;
{Variables for calculating program
finish time}


{*************************** I/O
Functions ***************************}

procedure OpenIn (var f: Text;
filename: string);

{Used to open specified files
containing information to be read into
the network}

begin {OpenIn}

  Assign(f, filename);
  Reset(f);

end;  {OpenIn}
```

```
procedure OpenOut (var f: Text;
filename: string);

{Used to open specified files to write
network output to}

begin {OpenOut}

  Assign(f, filename);
  Rewrite(f);

end;  {OpenOut}


{************************ Utility
Functions ************************}


function RandomOne: real;

{Returns a random number in the range -
1 to 1}

begin {RandomOne}

  RandomOne := Random * 2 - 1;

end;  {RandomOne}


function Random1To (topOfRange:
Integer): DataRange;

{Returns a random integer in the range
1 to topOfRange}

begin {Random1To}

  Random1To := Round(Random *
(topOfRange - 1)) + 1;

end;  {Random1To}


function LeadingZero (w : Word) :
String;

{Enables computation time to be
formatted correctly in output file}

var   s : String;

begin {LeadingZero}

  Str(w:0,s);

  if Length(s) = 1 then

    s:='0'+s;

  LeadingZero:=s;

end;  {LeadingZero}


{********************* Initialization
Functions *********************}
procedure InitParamSpec;

{Initialises the Parameter
Specification for the NN M.Alarc¢n 24-
11-00}

begin {InitParamSpec}

  ParamSpecFile := 'c:\psc.txt';
  OpenIn (fIn, ParamSpecFile);
```

```
  readln   (fIn, DataDirectory);
  readln   (fIn, NumInputs);
  readln   (fIn, NumOutputs);
  readln   (fIn, TrainPatterns);
  readln   (fIn, TestPatterns);
  readln   (fIn, InputPatterns);
  readln   (fIn, MaxIterations);
  readln   (fIn, NumHiddenNodes);
  readln   (fIn, sigma);
  Close  (fIn);

  EpochSize      := TrainPatterns;
{Number of training data patterns
considered in each epoch}
end;  {InitParamSpec}


procedure InitFilenames;
{Initialises tFilenames for the NN
M.Alarc¢n 7-12-00}

begin {InitFilenames}
  dataIn         := DataDirectory +
ImportAnal + 'trn' + ParamIndex +
'.txt'; {Input file containing training
data set}
  testDataIn     := DataDirectory +
ImportAnal + 'tst' + ParamIndex +
':txt'; {Input file containing test
data set}
  errorOut       := DataDirectory +
ImportAnal + 'err' + ParamIndex +
'.out'; {Output file for training and
test error}
  trainingOut    := DataDirectory +
ImportAnal + 'trn' + ParamIndex +
'.out'; {Output file for training
results}
  testOut        := DataDirectory +
ImportAnal + 'tst' + ParamIndex +
'.out'; {Output file for test results}
  weightsOut     := DataDirectory +
ImportAnal + 'wts' + ParamIndex +
'.out'; {Output file for weights
matrix}
  time           := DataDirectory +
ImportAnal + 'tim' + ParamIndex +
'.out'; {Output file for computation
time}

  inputDataFile := DataDirectory +
'input.txt';  {Input file containing
validation data set}
  netOutputfile := DataDirectory +
'output.out'; {Output file for
validation results}



end;  {InitFilenames}


procedure InitDataStrs (var d:
DataType; var t: TestDataType);

{Initialises the training and test data
structures on the heap}

var  m: DataRange;
     n: TestDataRange;

begin {InitDataStrs}

  for m := 1 to TrainPatterns do

    New(d[m]);

  for n := 1 to TestPatterns do

    New(t[n]);

end;  {InitDataStrs}


procedure DisposeDataStrs (var d:
DataType; var t: TestDataType);

{Disposes of the training and test data
structures}

var  m: DataRange;
     n: TestDataRange;

begin {DisposeDataStrs}

  for m := 1 to TrainPatterns do

    Dispose(d[m]);

  for n := 1 to TestPatterns do

    Dispose(t[n]);

end;  {DisposeDataStrs}


procedure InitData (var d: DataType;
var t: TestDataType);

{Reads in training and testing data
from the specified files}

var  m: DataRange;
     n: TestDataRange;
     i: InputRange;
     k: OutputRange;

begin {InitData}

  writeln;
  writeln('Reading in the training
data');
  writeln;
  OpenIn(fIn, dataIn);

  for m := 1 to TrainPatterns do

   begin {m}

    for i := 1 to NumInputs do

      read(fIn, d[m]^.x[i]);

    for k := 1 to NumOutputs do

      read(fIn, d[m]^.out[k]);

    readln(fIn);

   end;  {m}

  Close(fIn);
  writeln('Reading in the test data');
  OpenIn(fIn, testDataIn);

  for n := 1 to TestPatterns do

   begin {n}

    for i := 1 to NumInputs do

      read(fIn, t[n]^.x[i]);

    for k := 1 to NumOutputs do
```

```
      read(fIn, t[n]^.out[k]);

    readln(fIn);

   end;  {n}

  writeln;
  writeln(MaxIterations,' iterations
will now commence to generate network
weights');
  writeln;

  Close(fIn);

end;  {InitData}


procedure InitHiddenLayer (var inp:
DataType; var h: HiddenPEType);

{Set weights for each centre function,
ie. input to hidden layer weights}

var  i: InputRange;
     j: HiddenRange;

begin {InitHiddenLayer}

  for i := 1 to NumInputs do

    h.w[0, i] := 0.0;

  for j := 1 to NumHiddenNodes do

   for i := 1 to NumInputs do

     h.w[j, i] := inp[j]^.x[i];

end;  {InitHiddenLayer}


procedure InitOutputLayer (var o:
OutputPEType);

{Initializes the output weights to
random reals in the range -1 to 1}

var  j: HiddenRange;
     k: outputRange;

begin {InitOutputLayer}

  for k := 1 to NumOutputs do

    for j := 0 to NumHiddenNodes do

      o.u[k, j] := RandomOne;

end;   {InitOutputLayer}


{********************** Neural Net
Procedures **********************}


procedure NetForwardDel (var inp:
InputPEType; var h: HiddenPEType; var
o: OutputPEType; var delta:
DeltaErrorType);

{Calculates the output of the net for a
given input and also the error, delta}

var  net, val: real;
     i: InputRange;
     j: HiddenRange;
     k: OutputRange;

begin {NetForwardDel}

  {Hidden layer forward}

  for j := 1 to NumHiddenNodes do

   begin {j}

    net := 0;

    for i := 1 to NumInputs do

     net := net + sqr(inp.x[i] - h.w[j,
i]);

    h.z[j] := exp(-net/(2 *
sqr(sigma)));

   end;  {j}

  h.z[0] := 1; {Bias unit connected to
output layer}

  {Output layer forward}

  for k := 1 to NumOutputs do

   begin {k}

    val := 0;

    for j := 0 to NumHiddenNodes do

     val := val + o.u[k, j] * h.z[j];

    o.y[k] := val;
    delta[k] := inp.out[k] - o.y[k];

   end;  {k}

end;  {NetForwardDel}


procedure NetForward (var inp:
InputPEType; var h: HiddenPEType; var
o: OutputPEType; var
d1,d2:OutputOutType);

var  net, val: real;
     i: InputRange;
     j: HiddenRange;
     k: OutputRange;

begin {NetForward}

  {Hidden layer forward}

  for j := 1 to NumHiddenNodes do

   begin {j}

     net := 0;

     for i := 1 to NumInputs do

      net := net + sqr(inp.x[i] -
h.w[j, i]);

     h.z[j] := exp(-net/(2 *
sqr(sigma)));

   end;  {j}

  h.z[0] := 1; {Bias unit connected to
output layer}
```

```
  {Output layer forward}

  for k := 1 to NumOutputs do

   begin {k}

    val := 0;

    for j := 0 to NumHiddenNodes do

     val := val + o.u[k, j] * h.z[j];

    o.y[k] := val;
    d1[k]:= inp.out[k];
    d2[k]:= o.y[k];

   end;  {k}

end;  {NetForward}


procedure NetTrain (var inp:
InputPEType; var h: HiddenPEType; var
o: OutputPEType; newzeta: real);

{This procedure calculates an output
for any given input, compares the
predicted output
 with the given output, calculates the
associated error, delta, and updates
the output layer
 weights using the "delta rule"}

var  i: InputRange;
     j: HiddenRange;
     k: OutputRange;
     net, val, deltaJ: real;

begin {NetTrain}

  {Hidden layer forward}

  for j := 1 to NumHiddenNodes do

   begin {j}

    net := 0;

    for i := 1 to NumInputs do

     net := net + sqr(inp.x[i] - h.w[j,
i]);

    h.z[j] := exp(-net/(2 *
sqr(sigma)));

   end;  {j}

  h.z[0] := 1; {Bias unit connected to
output layer}

  {Output layer forward}

  for k := 1 to NumOutputs do

   begin {k}

    val := 0;

    for j := 0 to NumHiddenNodes do

     val := val + o.u[k, j] * h.z[j];

    o.y[k] := val;
    delta[k] := inp.out[k] - o.y[k];

   end;  {k}

  {Update output layer weights}

  for k := 1 to NumOutputs do

    for j := 0 to NumHiddenNodes do

      o.u[k, j] := o.u[k, j] + newzeta
* delta[k] * h.z[j];

end;  {NetTrain}


{************************ Display
Procedures ************************}


procedure DisplayError (var rms: real;
inD: DataType; tD: TestDataType; h:
HiddenPEType; o: OutputPEType; newzeta:
real);

{Calculates the Root Mean Square, RMS,
error for all the training and test
inputs}
{Displayed on user screen for
monitoring purposes and also written to
the specified file}

var  trErr, testErr, val, testRms:
real;
     del: DeltaErrorType;
     m: DataRange;
     n: TestDataRange;
     k: OutputRange;

begin {DisplayError}

  {Training data set calculations}

  trErr := 0;

  for m := 1 to TrainPatterns do

   begin {m}

    NetForwardDel(inD[m]^, h, o, del);
    val := 0;

    for k := 1 to NumOutputs do

     val := val + del[k] * del[k];

    trErr := trErr + val;

   end;  {m}

  rms := Sqrt(trErr/TrainPatterns);
{RMS error over all training patterns}

  {Test data set calculations}

  testErr := 0;

  for n := 1 to TestPatterns do

   begin {n}

    NetForwardDel(tD[n]^, h, o, del);
    val := 0;

    for k := 1 to NumOutputs do

     val := val + del[k] * del[k];

    testErr := testErr + val;
```

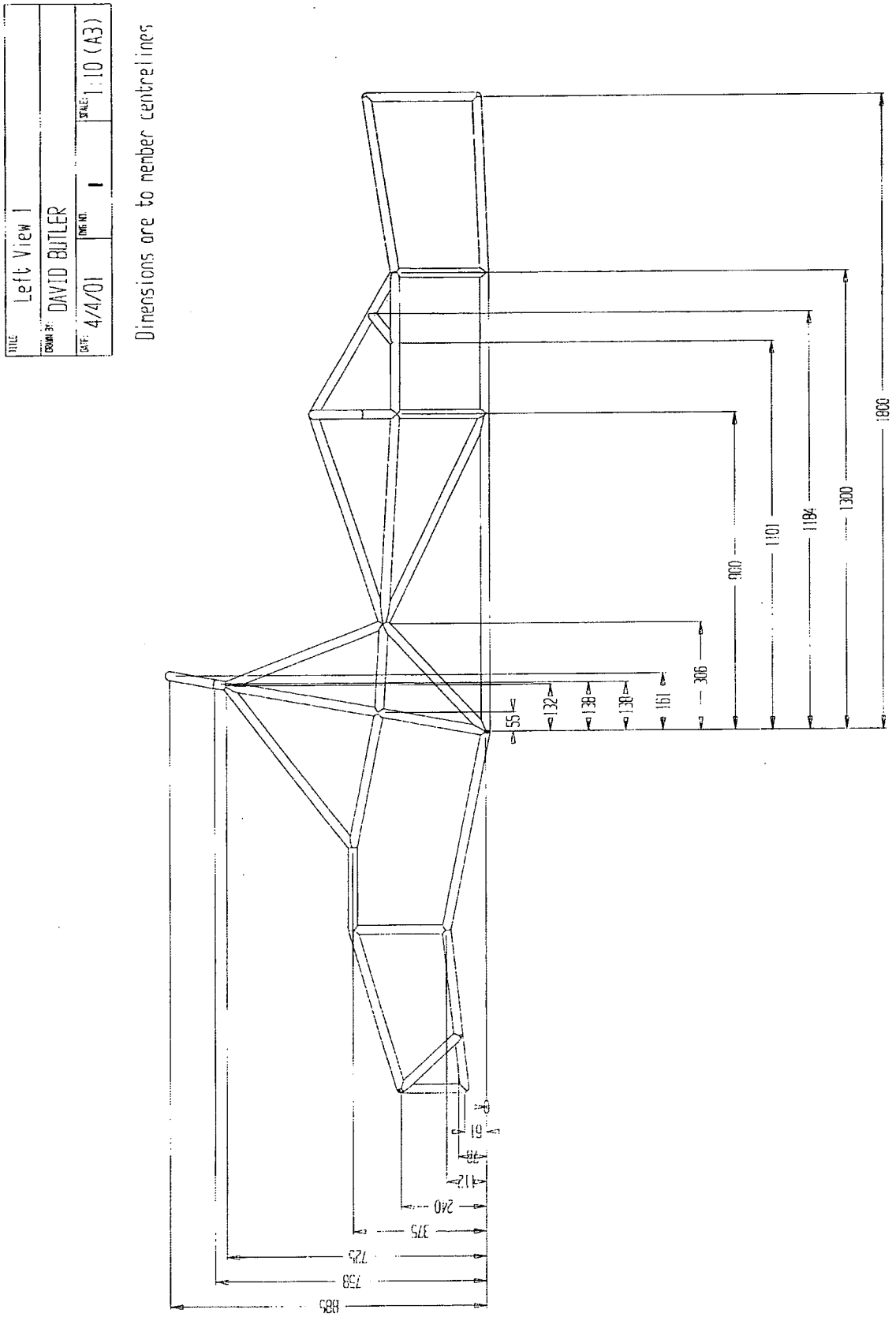
```
    end;  {n}

   testRms :=
Sqrt(testErr/TestPatterns); {RMS error
over all test patterns}

   writeln(fOut, iterations : 3, '  ',
rms : 5 : 5, '  ', testRms : 5 : 5);

end;  {DisplayError}


procedure WriteOutTrainingData (var
inD: DataType; var h: HiddenPEType; var
o: OutputPEType);

{Writes the predicted and actual
outputs from training to the specified
file}

var  curve1, curve2: OutputOutType;
     p: DataRange;
     k: OutputRange;

begin {WriteOutTrainingData}

   OpenOut(fOut, trainingOut);

   for p := 1 to TrainPatterns do

    begin {p}

     NetForward(inD[p]^, h, o, curve1,
curve2);

     for k :=1 to NumOutputs do

      write(fOut, curve1[k] : 7 : 5, '
');

     for k :=1 to NumOutputs do

      write(fOut, curve2[k] : 7 : 5, '
');

     writeln(fOut);

    end;  {p}

   Close(fOut);

end;  {WriteOutTrainingData}


procedure WriteOutTestData (var tD:
TestDataType; var h: HiddenPEType; var
o: OutputPEType);

{Writes the predicted and actual
outputs from testing to the specified
file}

var  curve1, curve2: OutputOutType;
     q: TestDataRange;
     k: OutputRange;

begin {WriteOutTestData}

   OpenOut(fOut, testOut);

   for q := 1 to TestPatterns do

    begin {q}

     NetForward(tD[q]^, h, o, curve1,
curve2);

     for k := 1 to NumOutputs do

      write(fOut, curve1[k] : 7 : 5, '
');

     for k := 1 to NumOutputs do

      write(fOut, curve2[k] : 7 : 5, '
');

     writeln(fOut);

    end;  {q}

   Close(fOut);

end;  {WriteOutTestData}


procedure WriteOutWeights (var h:
HiddenPEType; var o: OutputPEType);

{Writes the weight matrix to the
specified file}

var  i: InputRange;
     j: HiddenRange;
     k: OutputRange;

begin {WriteOutWeights}

   OpenOut(fOut, weightsOut);

   for j := 1 to NumHiddenNodes do

    begin {j}

     for i := 1 to NumInputs do

      write(fOut, h.w[j, i] : 8 : 5, '
');

     writeln(fOut);

    end;  {j}

   writeln(fOut);

   for j := 0 to NumHiddenNodes do

    begin {j}

     for k := 1 to NumOutputs do

      write(fOut, o.u[k,j] : 8 : 5, '
');

     writeln(fOut);

    end;  {j}

   Close(fOut);

end;  {WriteOutWeights}

{Procedures from here to "main program"
used for running option 2 of the
program}

procedure InitInputData (var d:
inputDataType);

{Reads input data from nominated file}

var  m: inputDataRange;
     i: InputRange;
```

```
      k: OutputRange;

begin {InitInputData}

   writeln('Reading in the input data');
   OpenIn(fIn, inputDataFile);

   for m := 1 to InputPatterns do

    begin {m}

     for i := 1 to NumInputs do

      read(fIn, d[m]^.x[i]);

     for k := 1 to NumOutputs do

      read(fIn,d[m]^.out[k]);

    end;  {m}

end;  {InitInputData}


procedure ReadInWeights (var
h:HiddenPEType; var o:OutputPEType);

{Reads the weights matrix from file
produced using option 1 of the program}

var  i,j,k : integer;

begin {ReadInWeights}

  OpenIn(fIn,weightsOut);

  for j := 1 to NumHiddenNodes do

   begin {j}

    for i := 1 to NumInputs do

      read(fIn, h.w[j, i]) ;

    readln(fIn);

   end;  {j}

  readln(fIn);

  for j := 0 to NumHiddenNodes do

   begin {j}

    for k := 1 to NumOutputs do

     read(fIn, o.u[k,j]);

    readln(fIn);

   end;  {j}

  Close(fIn);

end;  {ReadInWeights}


procedure WriteOutNetOutputData (var
tD: inputDataType; var h: HiddenPEType;
var o: OutputPEType);

{Calculates the output for any given
input using the weights developed
during training}

var  curve1, curve2: OutputOutType;
     q: inputDataRange;
      k: OutputRange;

begin {WriteOutTestData}

  OpenOut(fOut, netOutputFile);

  for q:=1 to InputPatterns do

   begin {q}

    NetForward(tD[q]^, h, o, curve1,
curve2);

    for k := 1 to NumOutputs do

     write(fOut, curve1[k] : 7 : 5, '
');

    for k := 1 to NumOutputs do

     write(fOut, curve2[k] : 7 : 5, '
');

    writeln(fOut);

   end;  {q}

   Close(fOut);

end;  {WriteOutTestData}

procedure InitInputDataStrs (var d:
inputDataType);

{Initialises the training and test data
structures on the heap}

var  m: inputDataRange;

begin {InitDataStrs}

  for m := 1 to InputPatterns do
   New(d[m]);

end;  {InitDataStrs}


{*************************** Main
Program ***************************}


var  answer: integer;
     count:    integer;
     stop:     integer;
     ni:       integer;
     buffer:   integer;
     newzeta: real;

begin {Main program}

  InitParamSpec; {Gets the Parameters
for the NN}

  clrscr;

  writeln;
  writeln;
  writeln('  Radial Basis Function
Neural Network');
  answer:=1;

  if (answer = 2) then

    begin {2}

     InitinputDataStrs (inputDataVar);
```

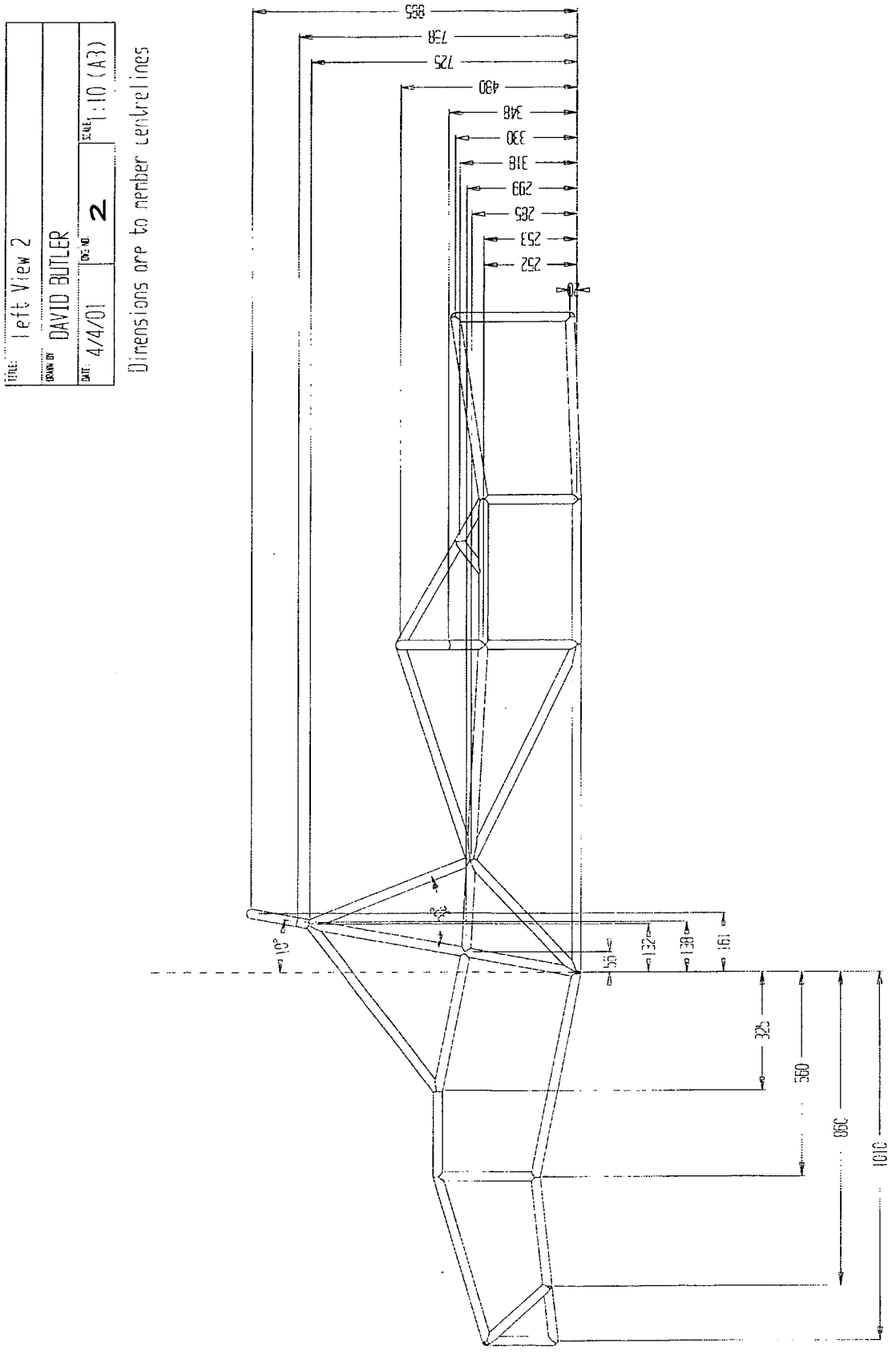
```
     ReadInWeights (HiddenNodes,
OutputNodes);
     InitInputData (inputDataVar);
     WriteOutNetOutputData
(inputDataVar, HiddenNodes,
OutputNodes);

    end   {2}

  else

  begin {1}
   ni:=NumInputs;
   stop:=ni*2;
   for count:=0 to stop do
   begin {count}
   if count=0 then
      begin
      ImportAnal:='';
      ParamIndex:='';
      InitFilenames;
      end;
   if (count>0) and (count<=ni) then
      begin
      ImportAnal:='pi';
      str(count,ParamIndex);
      NumInputs:=ni-1;
      InitFilenames;
      end;
   if count>ni then
      begin
      ImportAnal:='ci';
      buffer:=count-ni;
      str(buffer,ParamIndex);
      NumInputs:=ni-1;
      InitFilenames;
      end;
     GetTime (h1, m1, s1, hund1);

     {Initialize the training and test
data sets}

     InitDataStrs (inData, testData);
     InitData (inData, testData);
     OpenOut (fOut, errorOut);

     {Initialise output layer weights
using random values, set hidden layer
weights}

     randSeed := dataSeed;
     InitHiddenLayer
(inData,HiddenNodes);
     InitOutputLayer (OutputNodes);

     {A 'for' loop is used to fix the
total number of iterations}

     {Reset the random seed so that the
calling sequence can be contolled}

     randSeed := calcSeed;
     newzeta := zeta;

     for iterations := 1 to
MaxIterations do

      begin {iterations}

       for q := 1 to EpochSize do

        begin {q}

         inputData :=
InData[Random1To(TrainPatterns)]^;
         NetTrain(inputData,
hiddenNodes, outputNodes, newzeta);

        end;  {q}

       {Display RMS error for each
iteration}

       DisplayError (rmsError, InData,
testData, hiddenNodes, outputNodes,
newzeta);

       {Decrease delta rule constant
over the iterations range}

       newzeta := decrate * newzeta;

       if newzeta < 0.1 then

        newzeta := 0.1

       else

        newzeta := newzeta;

      end;  {iterations}

     Close(fOut);
     WriteOutTrainingData (inData,
hiddenNodes, outputNodes);
     WriteOutTestData (testData,
hiddenNodes, outputNodes);
     WriteOutWeights (hiddenNodes,
outputNodes);
     DisposeDataStrs (inData,
testData);
     GetTime (h2, m2, s2, hund2);
     OpenOut (fOut, time);
     writeln(fOut, 'Start time  :-
',LeadingZero(h1),':',LeadingZero(m1),'
:',LeadingZero(s1),':',LeadingZero(hund
1));
     writeln(fOut, 'Finish time :-
',LeadingZero(h2),':',LeadingZero(m2),'
:',LeadingZero(s2),':',LeadingZero(hund
2));
     Close(fOut);
     end; {count}
    end;  {1}
    Assign(fIn,'c:\psc.txt');
    Erase(fIn);

end.  {Main program}
```

Appendix B
Frame Specifications

B-1: Left view of frame (1st set of dimensions)

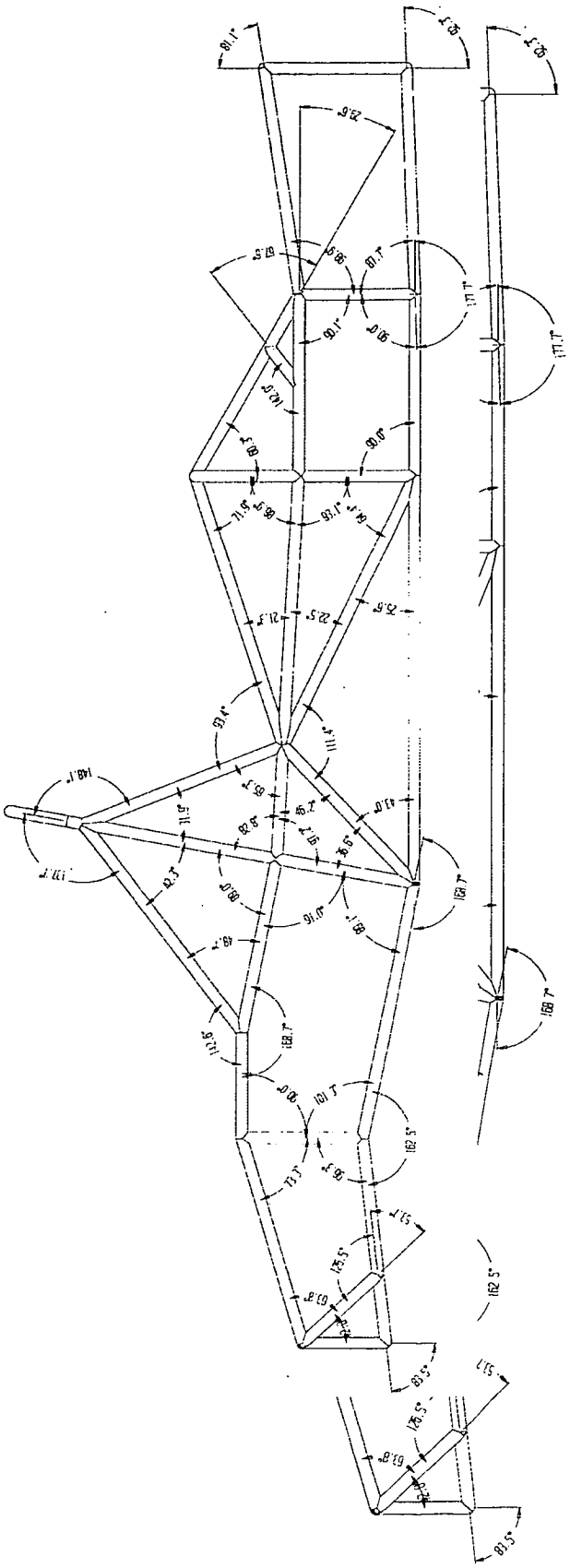


B-2: Left view of frame (2nd set of dimensions)



B-3: Left view of frame (member angles)

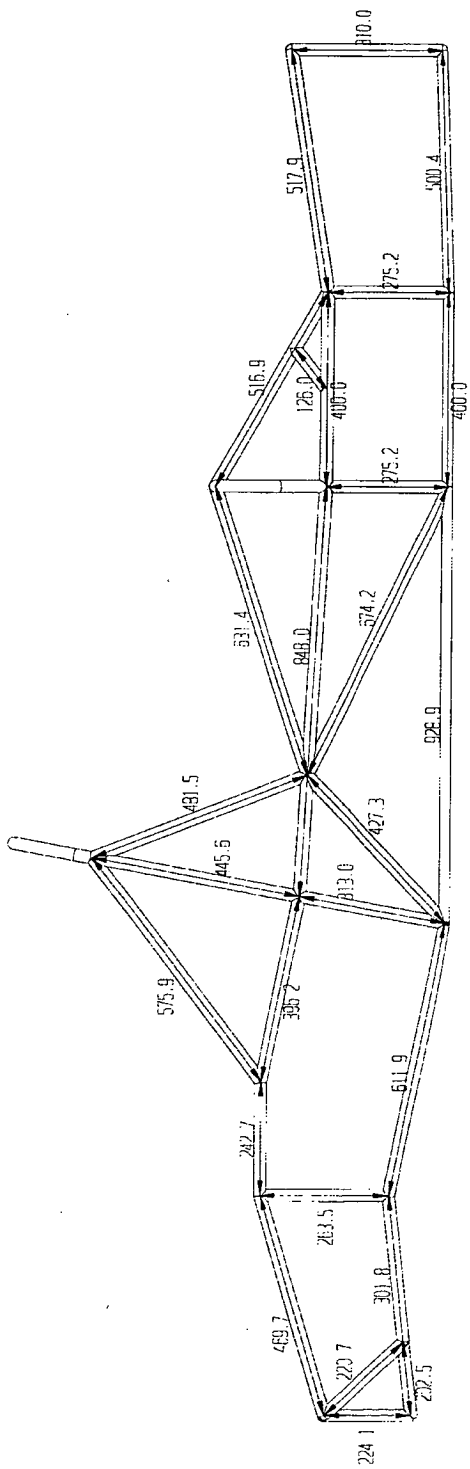
TITLE: Left View with Angles			
DRAWN BY: DAVID BUTLER			
DATE 4/4/01	DWG NO. 3	SCALE: 1:10 (A3)	



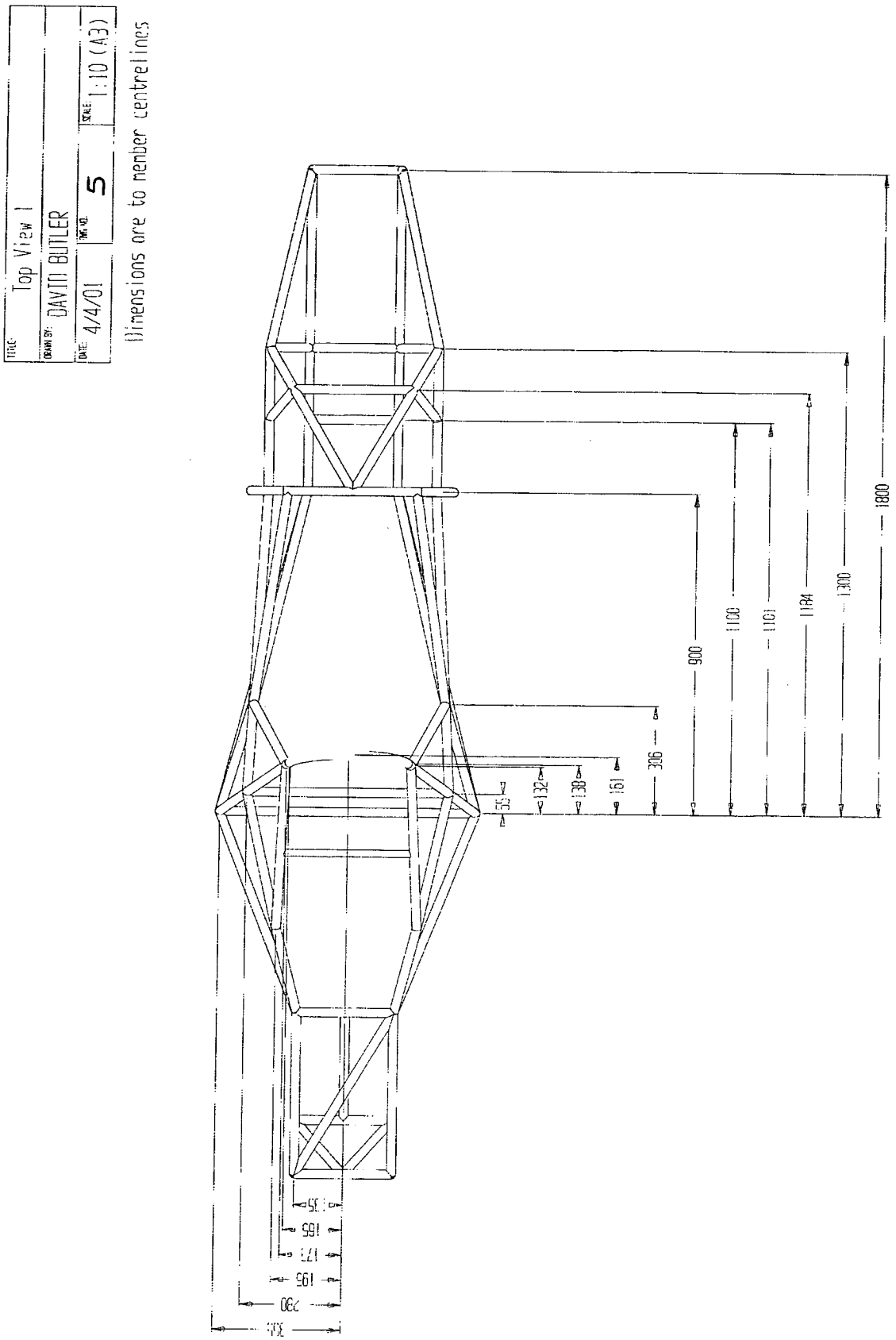
B-4: Left view of frame (three dimensional member lengths)

TITLE: Left View - 3D Member Dim.			
DRAWN BY: DAVID BUTLER			
DATE: 4/4/01	DWG NO 4	SCALE: 1:10 (A3)	

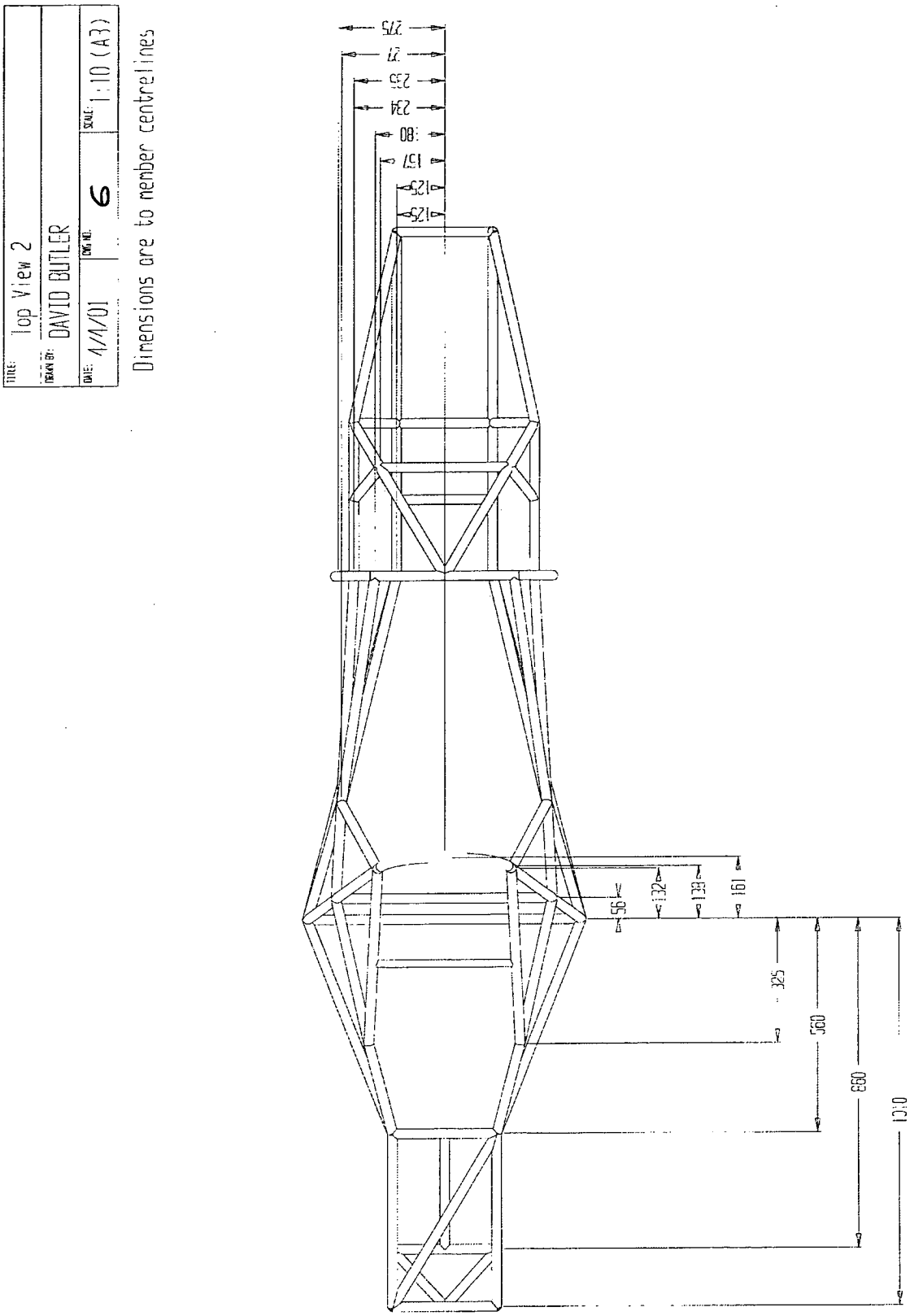
All member lengths are 3D



B-5: Top view of frame (first set of dimensions)

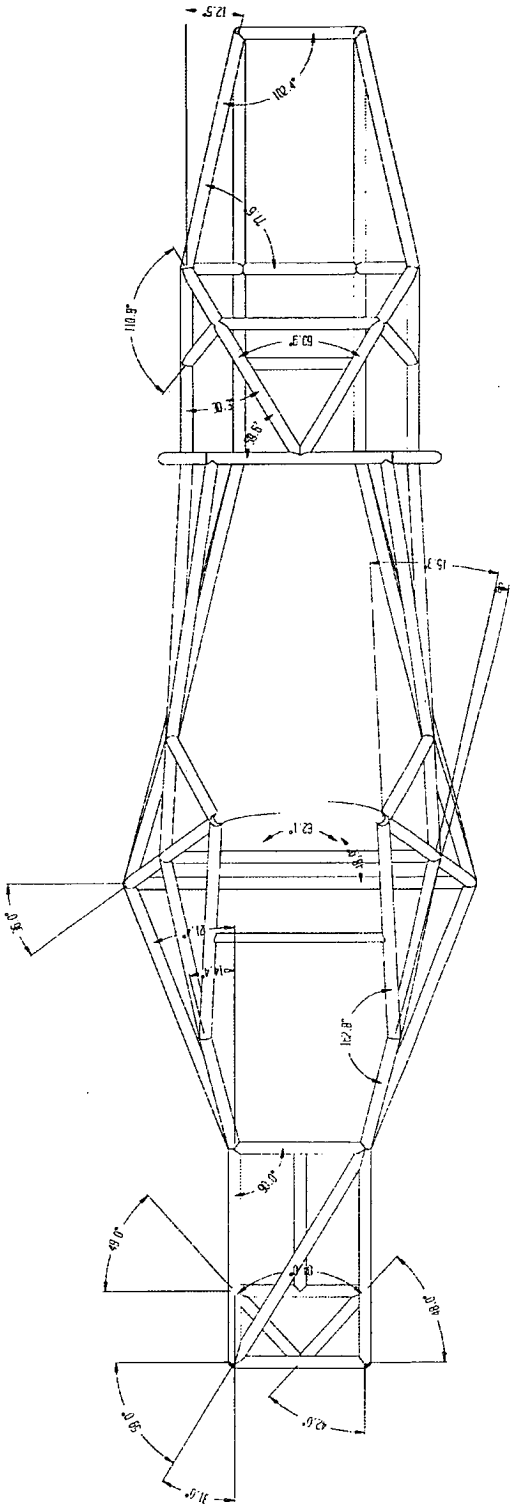


B-6: Top view of frame (second set of dimensions)

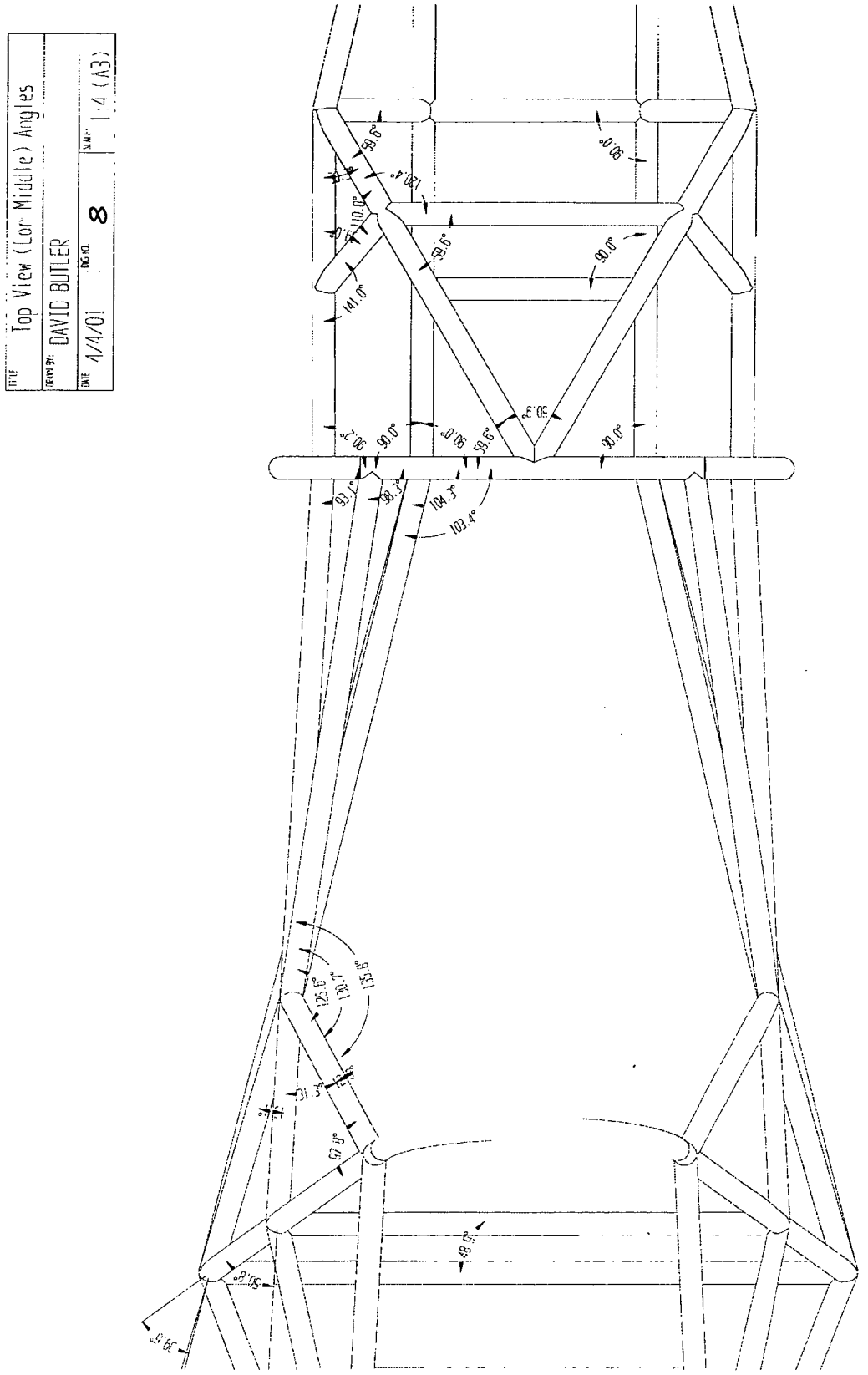


B-7: Top view of frame (member angles)

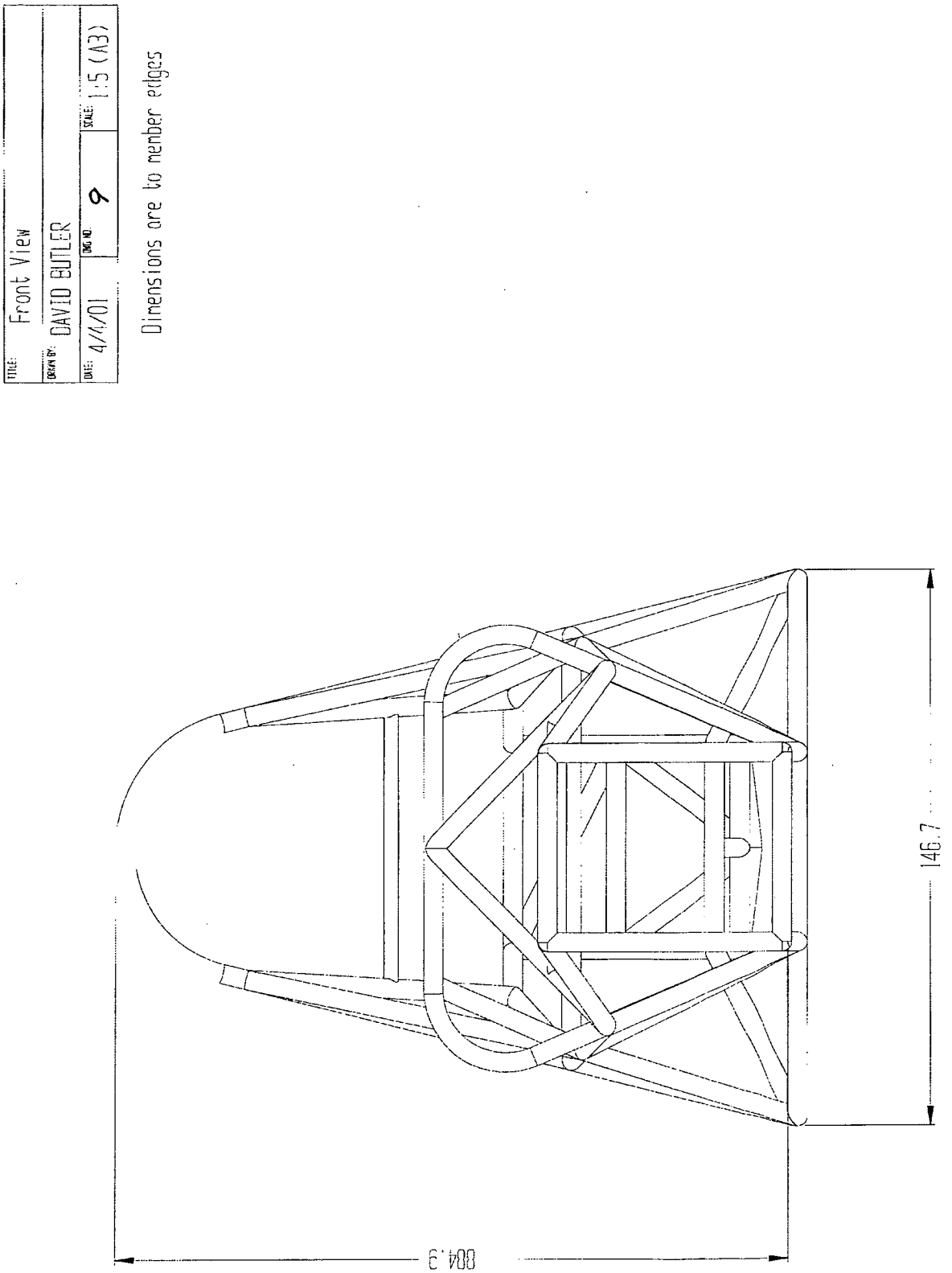
TITLE: Top View Angles			
DRAWN BY: DAVID BUTLER			
DATE: 4/4/01	DWG NO: 7	SCALE: 1:10 (A3)	



B-8: Top view of frame (member angles of mid-section)

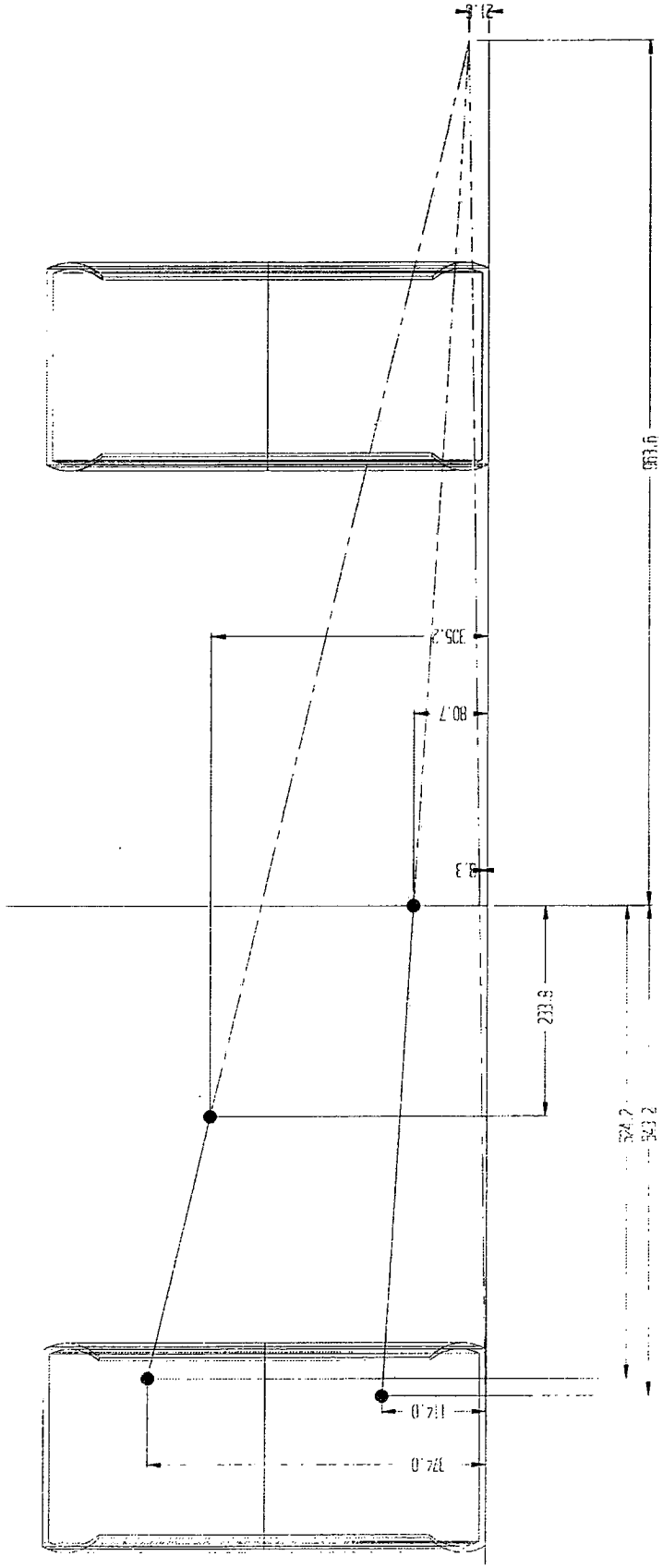

TITLE Top View (Cor Middle) Angles
DRAWN BY DAVID BUTLER
DATE 1/1/01
DWG NO. 8
SCALE 1:4 (A3)
90.0°
90.0°
90.0°

B-9: Front view of frame

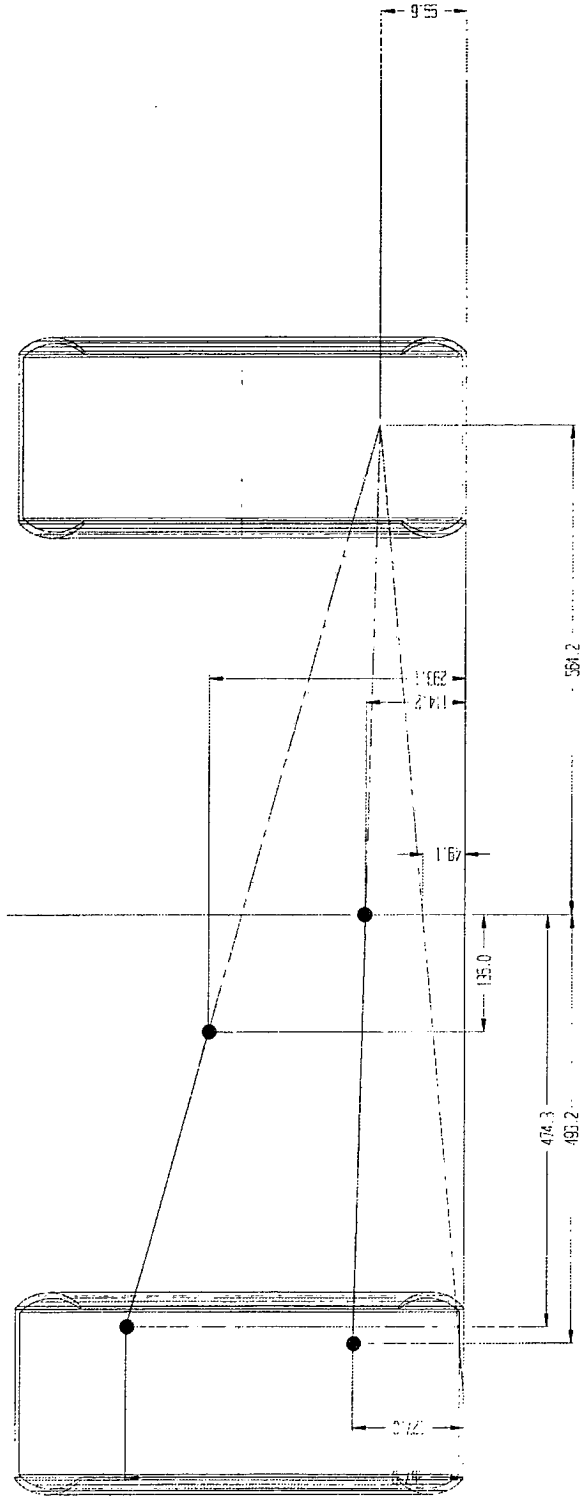


Appendix C
Suspension Specification

C-1: Front suspension geometry



C-2: Rear suspension geometry



C-3: Suspension spring specifications

RST 58

Coil spring with internal hydraulic floating piston

A6061-T6 aluminum body

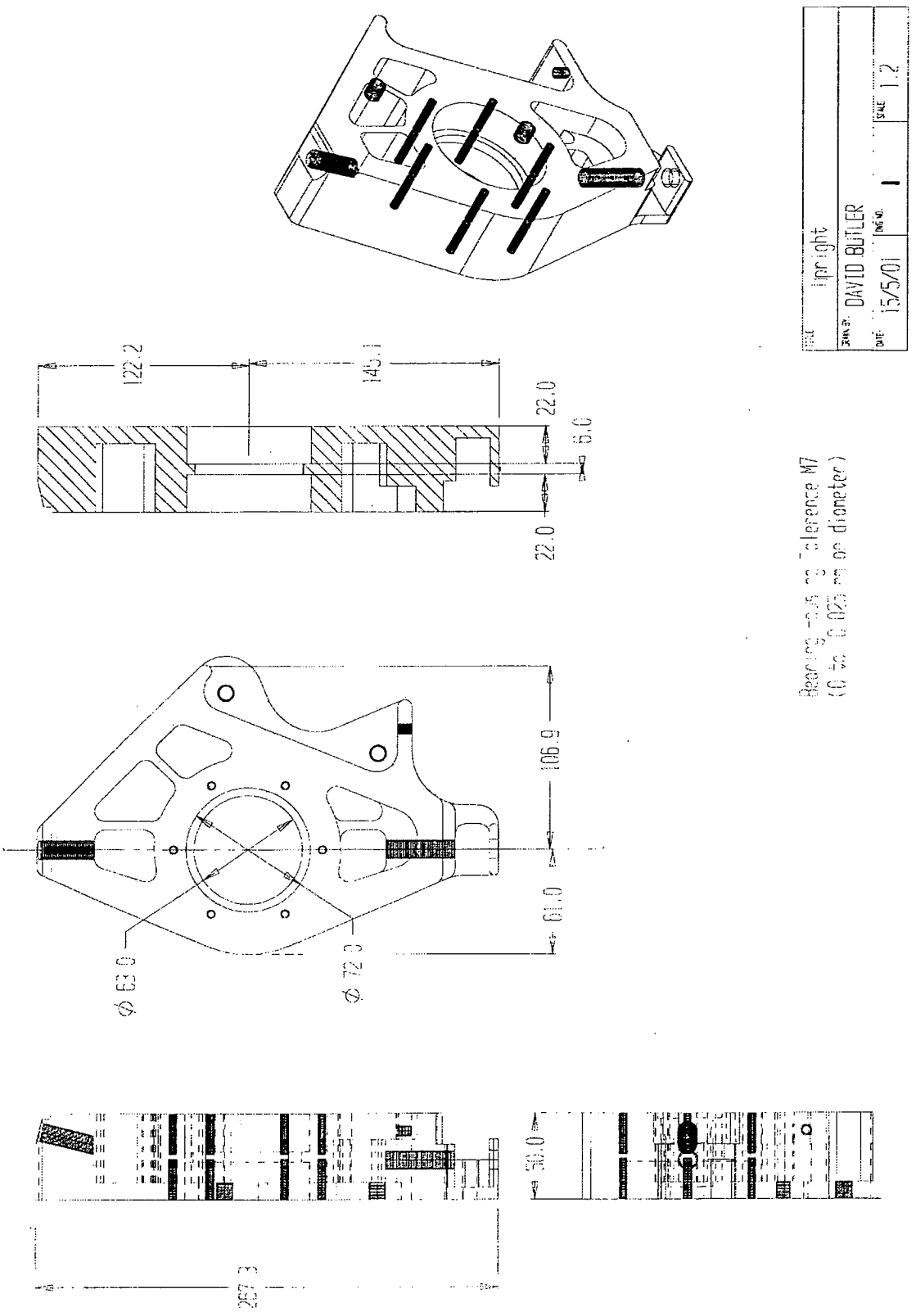
Spring pre-load adjustment

Spring standard color : black

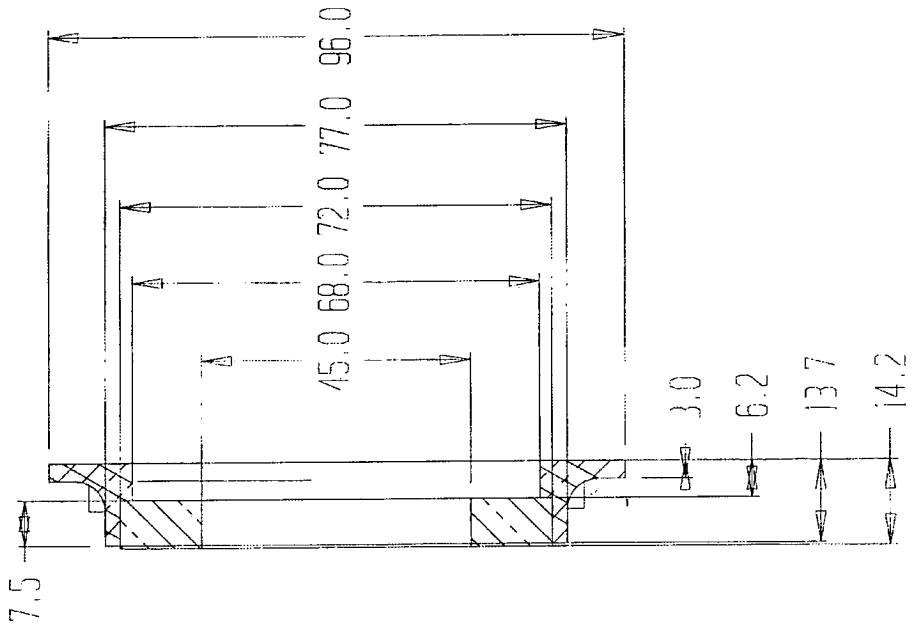
	58
COMPRESS DAMPING ADJUSTMENT	YES
COMPRESS LOCKOUT	YES
REBOUND DAMPING ADJUSTMENT	YES
EYE TO EYE LENGHT	155mm~190mm
TRAVEL	26mm~37mm

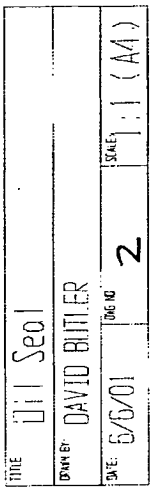
Appendix D
Wheel Assembly Specifications

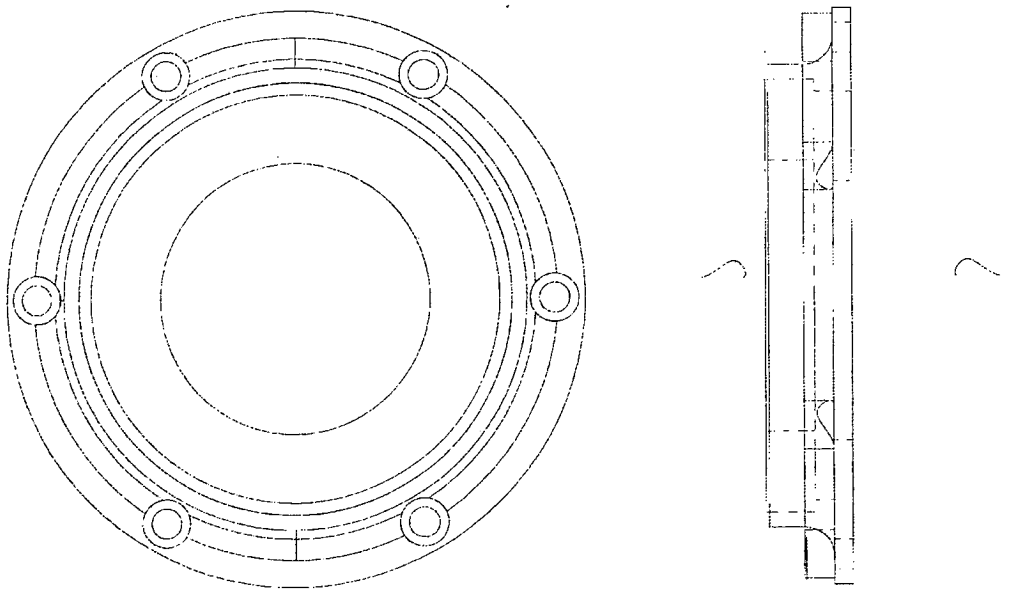
D-1: Upright dimensions



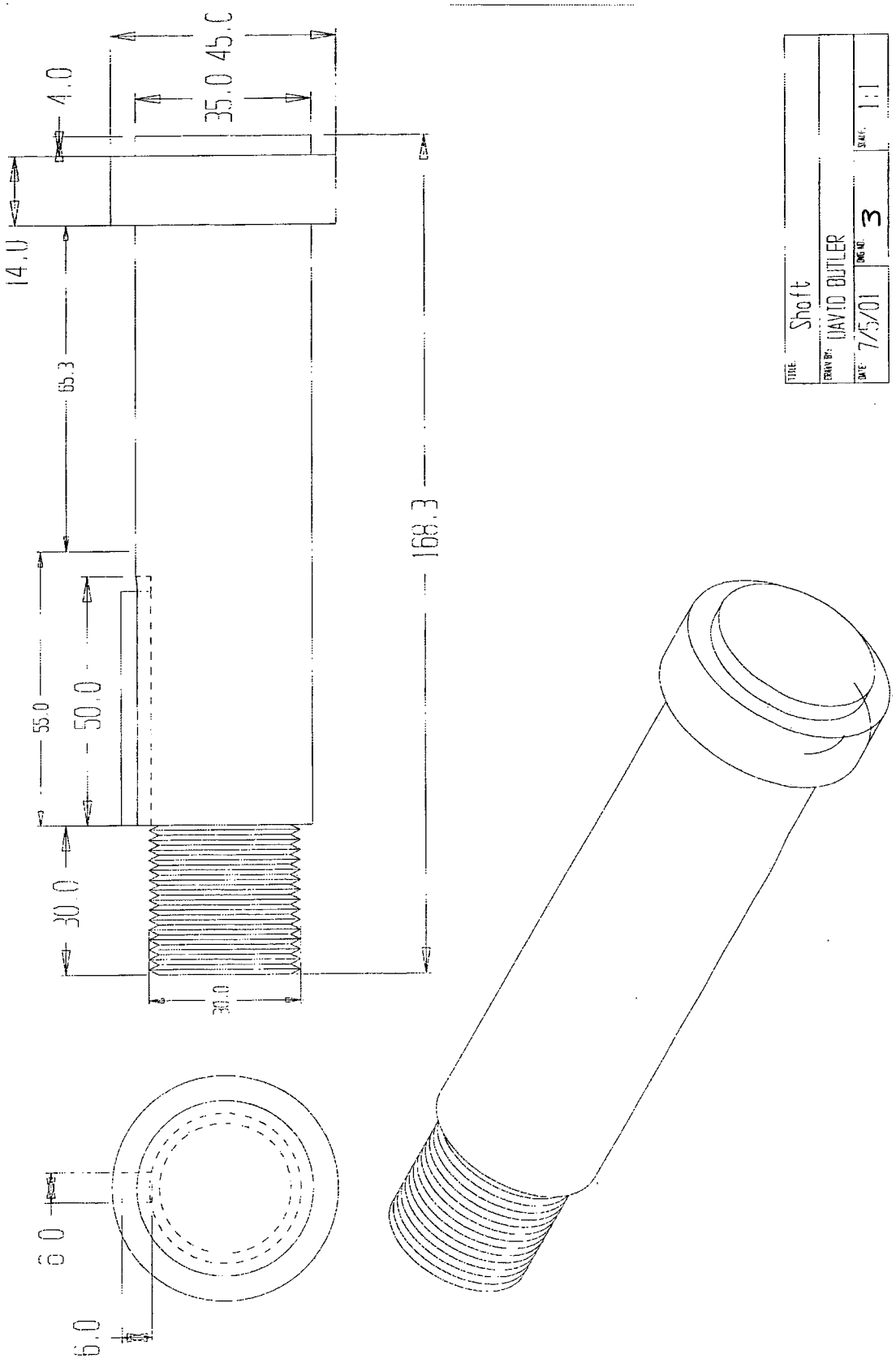
D-2: Oil seal dimensions



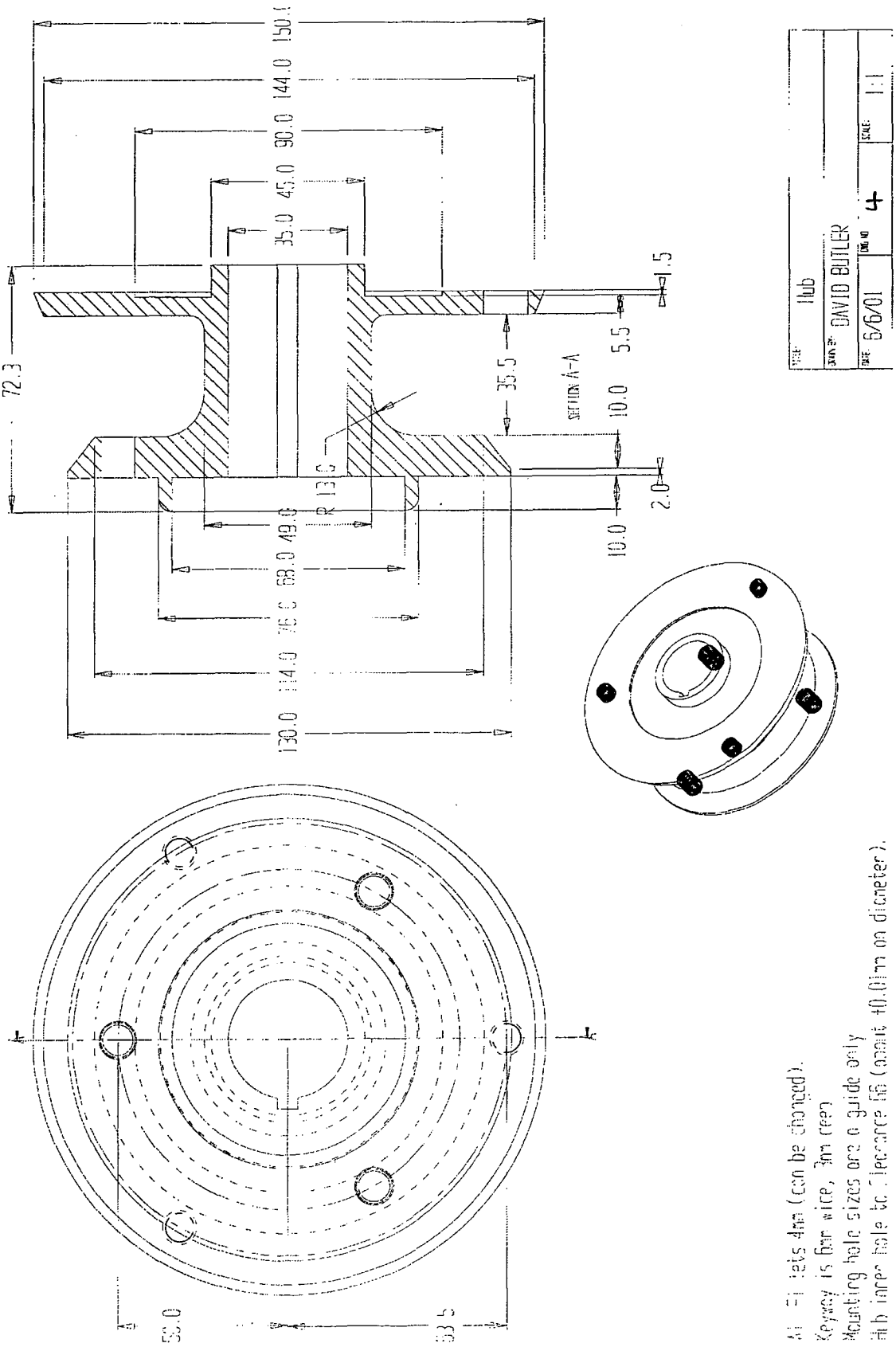




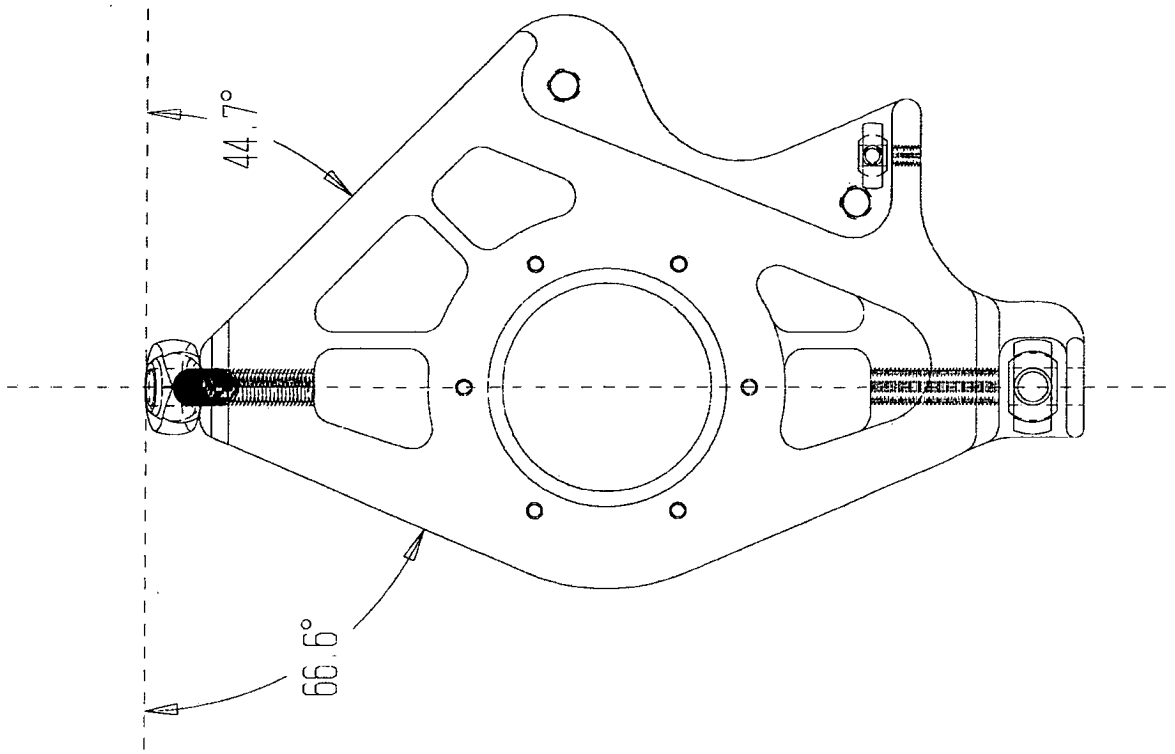
D-3: Stub axle dimensions

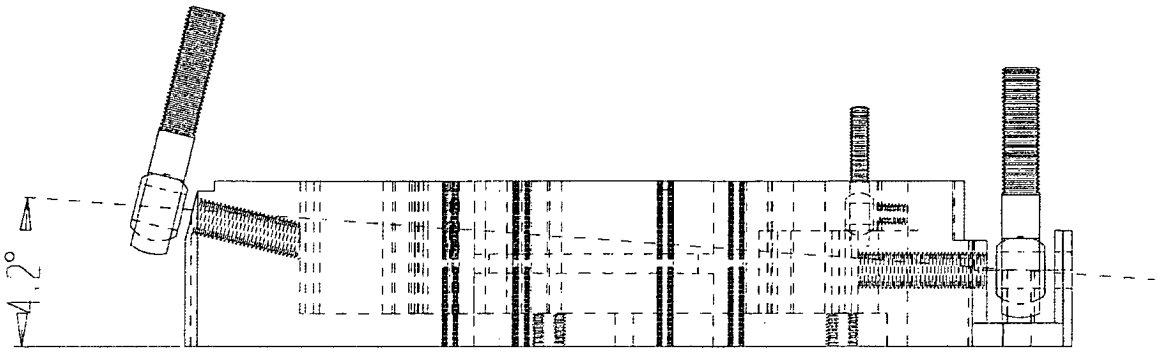


D-4: Wheel hub dimensions



D-5: King pin and caster reference angles



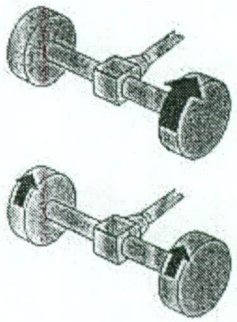


Appendix E
Drivetrain Specifications

Quaife Automatic Torque Biasing Differential for Added Traction

Design

The Quaife ATB Differential is designed to prevent the complete loss of drive that occurs with a conventional differential when one wheel slips.



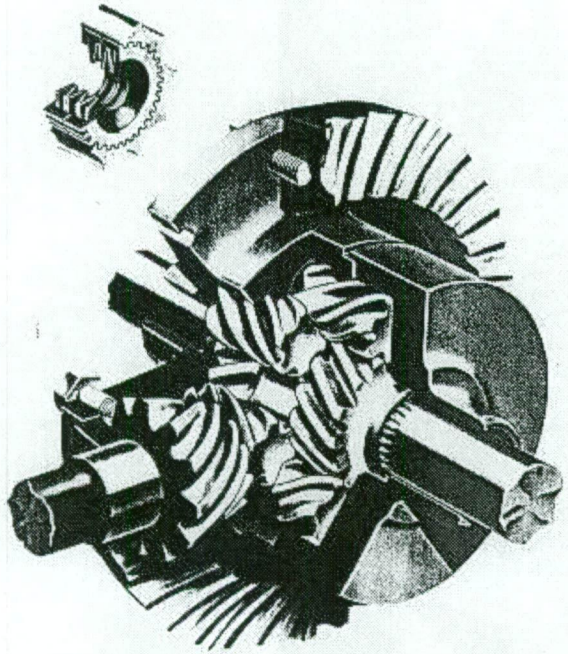
Whilst requiring some torque in the slipping wheel the Quaife unit is progressive in action but never locks - controlled power is transmitted to all the driving wheels. Although ideally suited to high-powered front-wheel drive systems, Quaife ATB differentials are used in rear and four-wheel drive vehicles where optimum traction is required. The four-wheel drive layout includes a centre differential as well as one in each axle.

The torque capacity of the Quaife unit is increased or decreased by varying the helix and pressure angles of the gear teeth. A combination is available to meet user requirements varying from Formula One racing to high mileage Ambulance fleets.

The operation is automatic, normal axle lubrication is retained and the unit is interchangeable with the conventional differential.

Operation

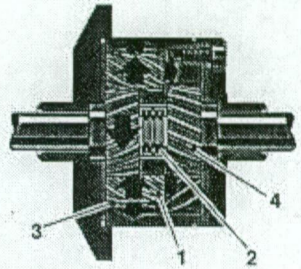
The Quaife Differential is an automatic gear-operated Torque Biasing Differential. Sets of floating helical gear pinions (1) mesh to provide the normal speed differential action. To pre-load the gear packs there is a selection of centre spring discs (2) available. In the event of wheel



slip torque bias is generated by the axial and radial thrusts (red) of the pinions in the pockets (3). The resultant friction force enables the driving road wheel and sun-gear (4) to transmit a greater proportion of the torque. This effect is progressive, but at no stage does the differential lock solid. Hence the inherent safety of the Quaife Automatic Torque Biasing Differential.

Fitting and Maintenance

Installation is identical to the normal differential with bearing pre-loads and pinion mesh being restored to the original manufacturers' recommendation. Servicing of the unit is simple as all



KEY
1 Helical gear pinions
2 Centre spring discs
3 Pocket
4 Sun Gear

gear pinions are free fitting and normal final drive lubrication oils are retained. Due to the internal design of the Quaife differential, all driving wheels must be elevated when servicing brakes, tyres, etc.

Applications

Quaife differentials are used in all forms of motor sport from circuit to rallying in two and four-wheel drive systems. A wide variety of emergency vehicles, where all-weather mobility is essential, also use Quaife differentials. Major users being Ambulance, Police, HM Coastguard, MOD, Forestry Commission and Public Utilities where the addition of a Quaife ATB differential improves vehicle handling and stability without compromising service life or operating costs. The benefits being available all year round whatever the traction conditions.

Company Profile

R.T. Quaife Engineering (established 30 years in UK) manufactures racing gear boxes (car and motor cycle), differentials, steering components and drive line power take-offs. The Differential design is patented and the Quaife name is a registered mark.

R. T. Quaife Engineering Ltd
Vestry Road, Otford, Sevenoaks, Kent.
TN14 5EL. England
Tel: 01732 741144
Fax: 01732 741555

E-2: Quaife ATB differential operation

Operation

The Quaife Differential is gear operated and therefore requires no plates which may wear or break. The unit is smooth in use and requires no special lubricants.

Designed & Manufactured in England.

SPECIFICATION

Design

The Quaife Differential is designed to power both wheels and control loss of drive. The differential provides constant and infinitely variable drive, traction being transferred from the spinning wheel to the static wheel automatically without the use of the normal friction pads in other designs. The operation is fully automatic and requires no manual control.

The unique design offers full maximum traction, improves handling and steering, and puts the power where it is needed most. With all the gears being the helical type, the helix and pressure angle of the gear teeth can be varied to increase or decrease the torque capacity.

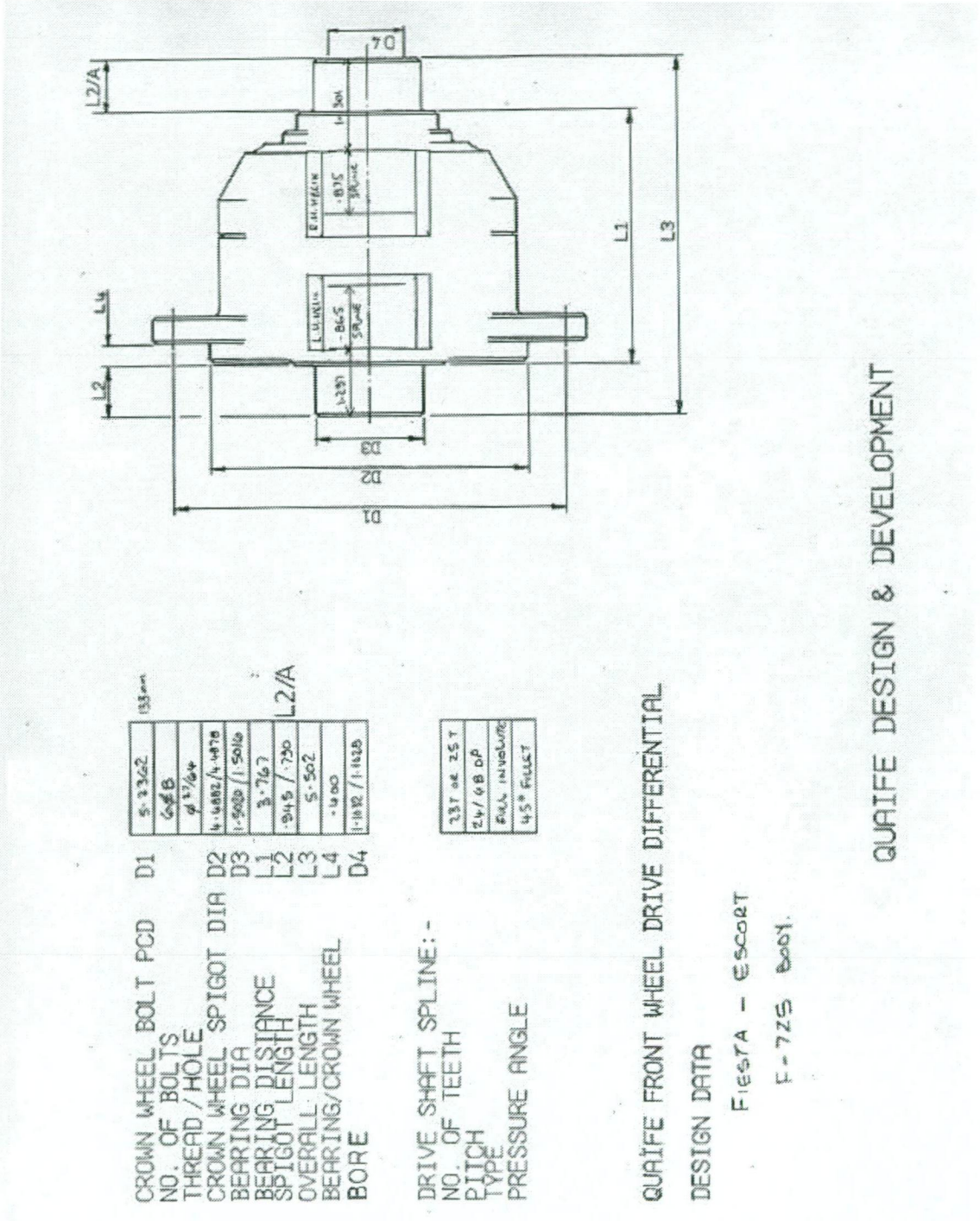
Suitability

The differential is ideally suited to four wheel drive applications, as well as competition vehicles. Can also be used in all four wheel drive units, both front and rear. Even when fitted to front wheel drive vehicles, there is no adverse resistance to the steering.

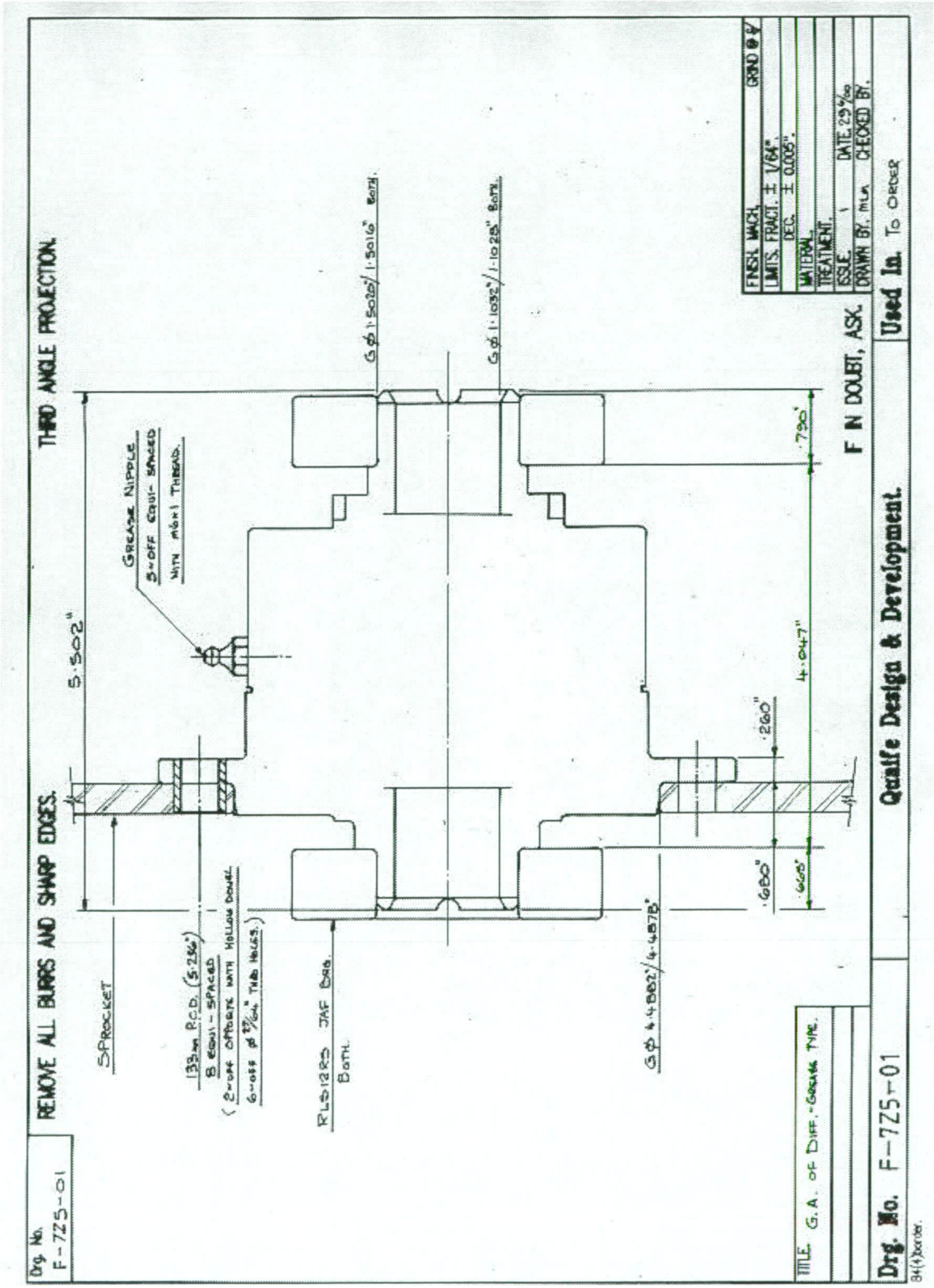
Fitting and Maintenance

Fitting the Quaife Differential is the same as installing the standard differential unit. Any maintenance can be carried out by a competent mechanic and no special tools are required.

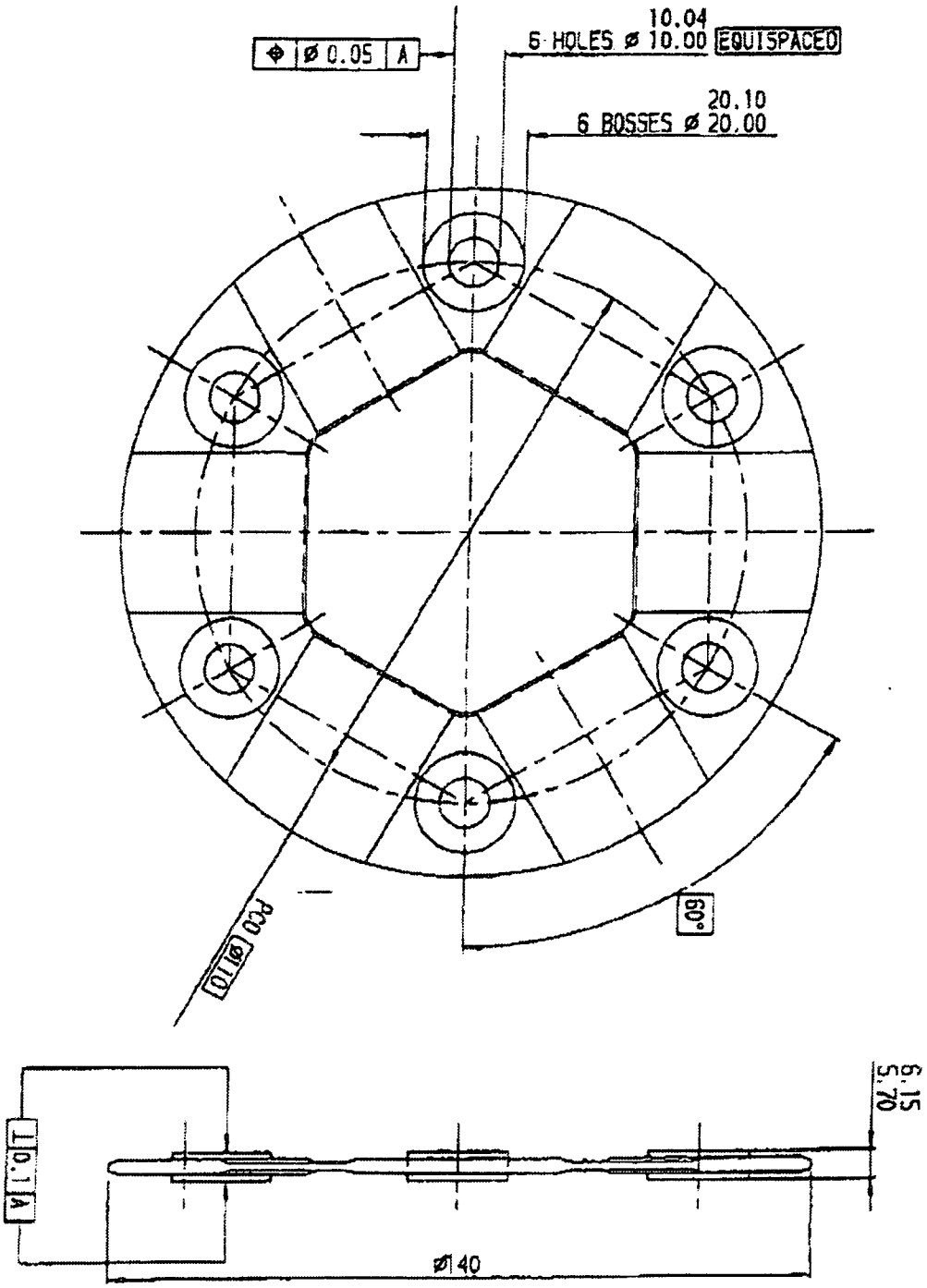
E-3: Quaife ATB differential specifications



E-4: Quaife ATB differential dimensions



E-5: Composite disc dimensions

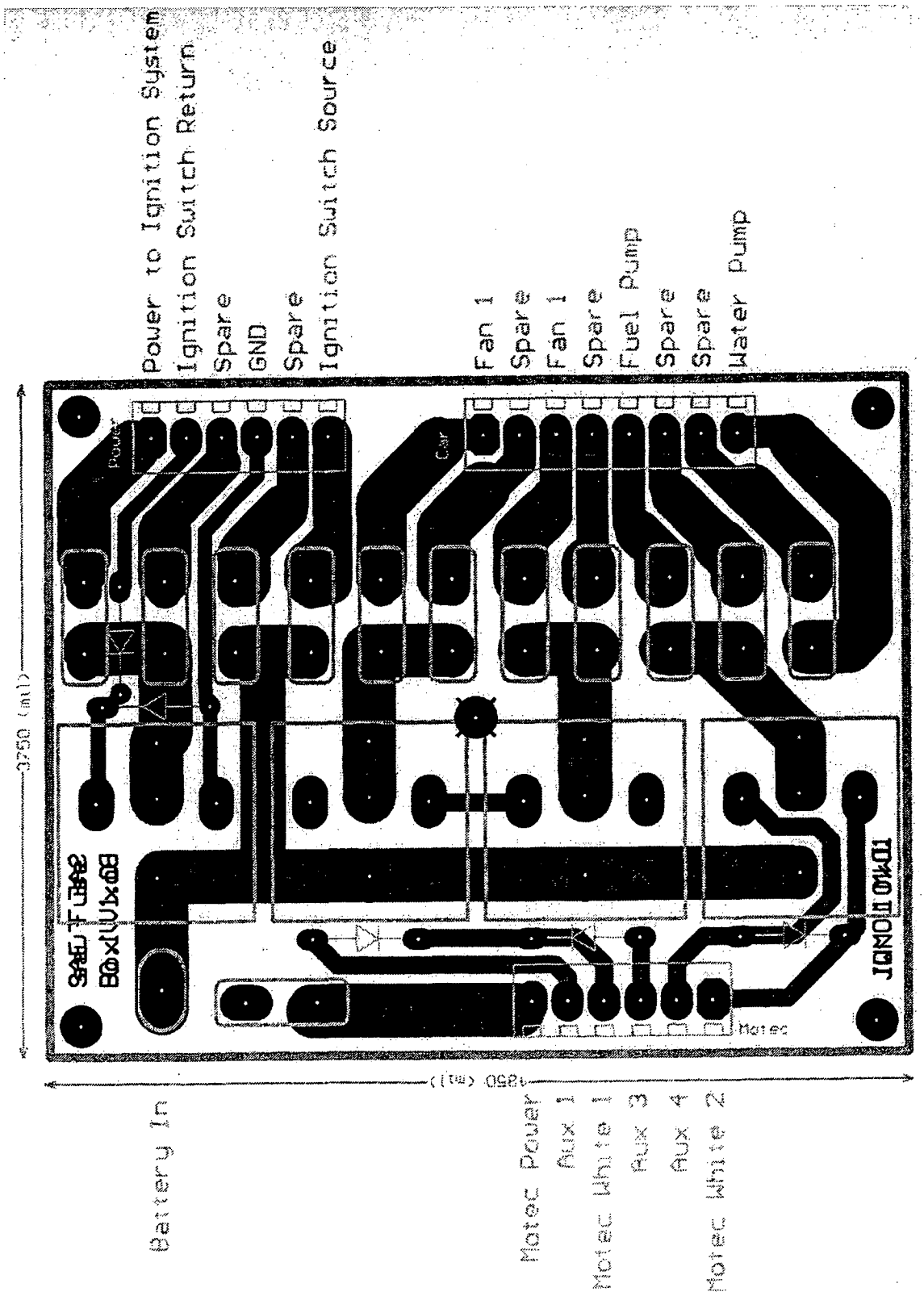


Appendix F
Electrical Systems Specifications

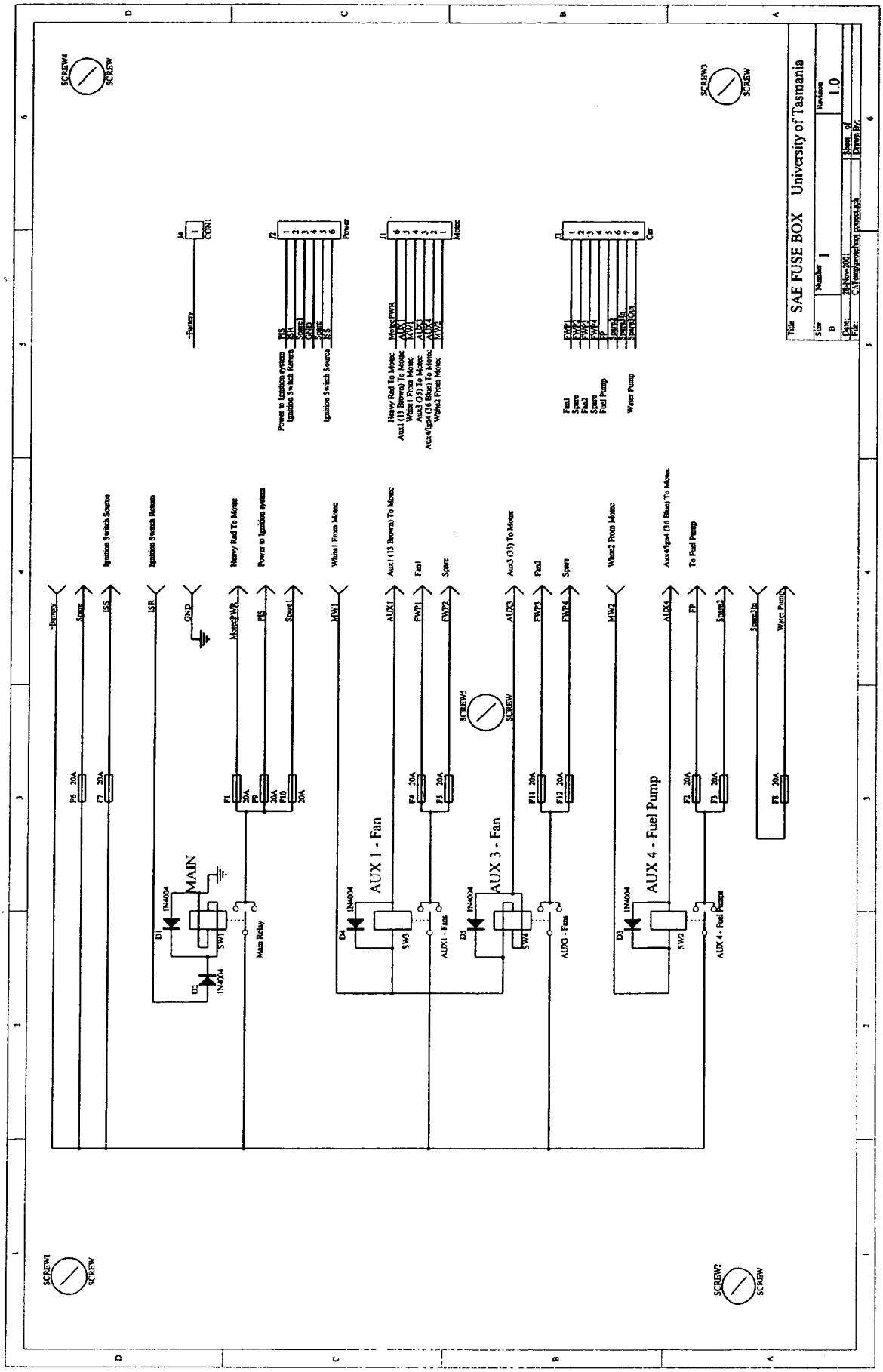
F-1: Electrical system general specifications

Battery	
Type	Gel cell
Capacity	15 Ah
Voltage	12.6 V
Charging System	
Type	Three Phase AC
Alternator output voltage	45 V
Stator coil resistance	0.2-0.6 Ohm
Charging Voltage	14 – 15 V
Electric Starter System	
Starter motor	12 V
Brush length	12mm

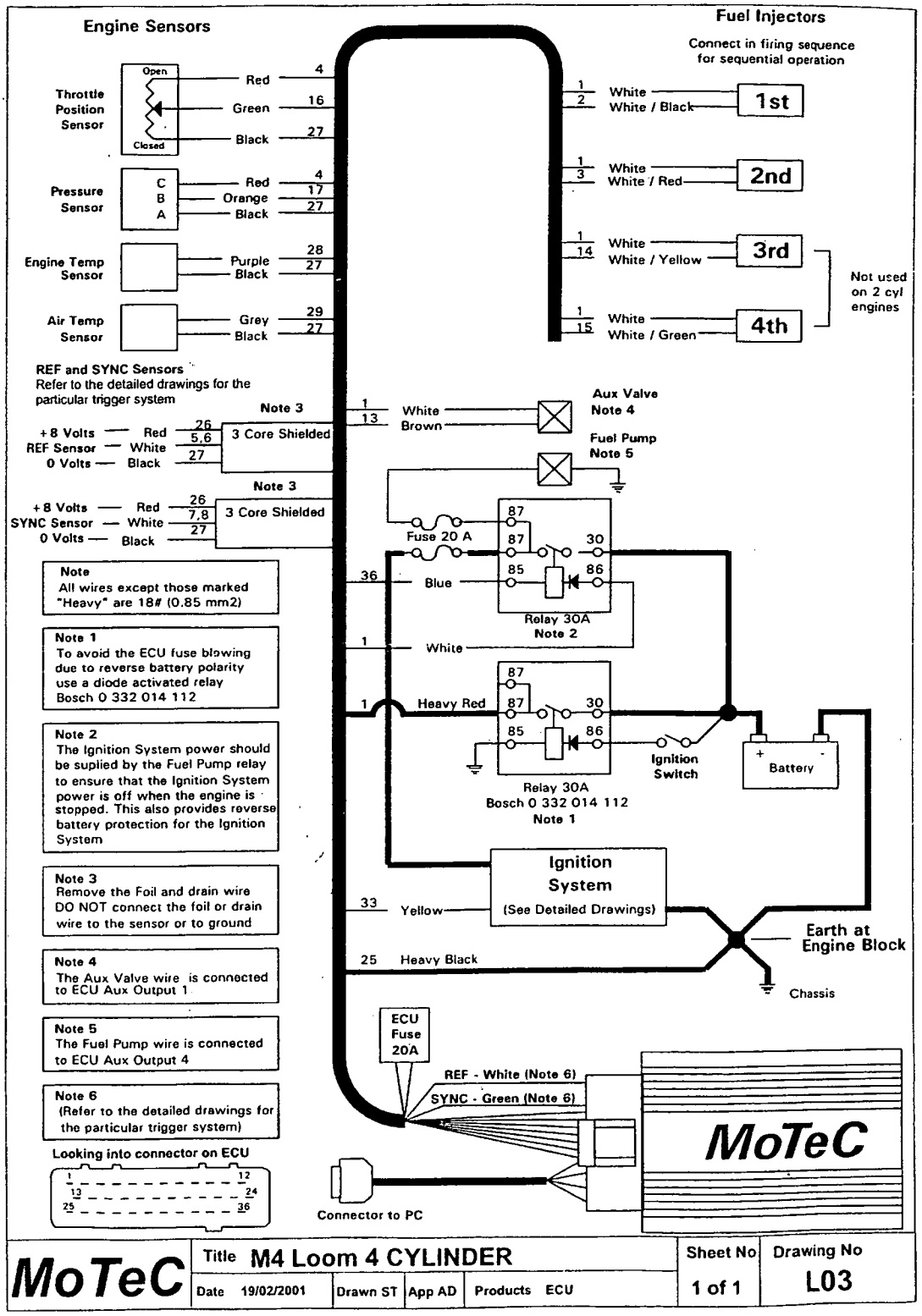
F-3: Fuse box circuit layout



F-2: Fuse box wiring diagram



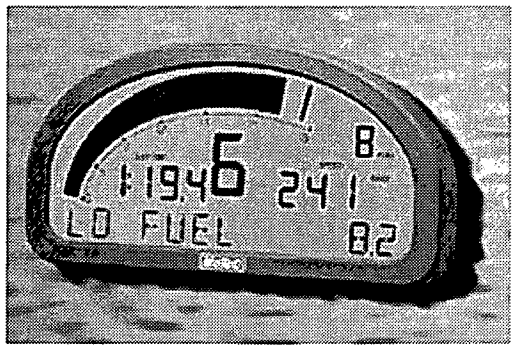
F-4: ECU Wiring Diagram



Appendix G
Sensor Specifications

G-1: Advanced dash logger - description

MoTeC Advanced Dash Logger (ADL)



The MoTeC Advanced Dash Logger (ADL) is a fully featured and self contained, programmable logger. The key difference between the MoTeC ADL and other products is its flexibility to be adapted to any application.

Many vehicle, marine and industrial applications require separate products to perform the logging, controlling and displaying. However, the MoTeC ADL offers seamless integration of all three functions.

All aspects of the ADL are fully configurable, including which sensor is connected to which input, what to log, how fast to log it, which channels to display, warning alarms and control outputs.

The MoTeC ADL uses a high speed 32 bit microprocessor and incorporates a 79 pin autosport connector. The ADL is built to internationally recognised quality and manufacturing standards and is back by a full 2 year worldwide warranty.

.1.1.1.1 Data Logging

Data logging allows for readings taken from Analog, Digital, Serial, CAN or Calculated channels, to be stored in the ADL for later analysis on a computer. The ADL uses permanent non-volatile Flash memory. Data memory may be unloaded at very high speeds (approx. 19 seconds per Mbyte). Different logging options allow 384k, 1MB or the full 8MB to be accessed.

The ADL can store channels at up to 1000 times per second per channel, this can be individually set for each channel. Four logging modes may run concurrently (Normal, Fastest Lap and two Burst Modes) each with selectable start and stop logging parameters. Memory can operate in stack or circular buffer mode.

.1.1.1.2

.1.1.1.3 Analog and Digital Channels

In total the ADL can accommodate over 200 channels derived from any mixture of Analog, Digital, RS232 Serial and CAN bus data channels.

The ADL directly supports up to 28 analog inputs, 12 digital/speed inputs, 8 auxiliary outputs and 2 high accuracy Wideband Lambda (Air/Fuel ratio) inputs.

The analog channels sample at up to 1000 samples per second per channel, with a measurement range of 0 to 15 VDC.

Digital inputs are used for state monitoring, counting and pulse width measurement. They accept switch, logic, open collector (Hall Effect), or magnetic signals.

The auxiliary outputs can be configured to operate as simple off/on outputs, duty cycle control or frequency based outputs.

.1.1.1.4

Serial Communications

The RS232 serial port is programmable up to 115k baud and can be used as either a telemetry data output port or serial data input port.

As a telemetry port; devices such as Modems, GSM & Satellite Phones, Radio Modems etc. can be connected to facilitate remote communication.

As a serial data input port; serial communication devices can be connected for displaying and logging purposes. These include Engine Management Systems (MoTeC and other), bar code devices, keypads, GPS Systems or other serial communications devices.

.1.1.1.5

Display

The MoTeC ADL display is a high contrast, high temperature, custom designed reflective LCD. Its unique design makes it viewable in direct sunlight or artificial light.

The display has 3 modes of operation, where each mode is fully programmable and independent of the other. Each mode may be selected by pressing a button or activated by a condition.

The 70 segment bar graph display is programmable to display any channel, with an optional peak hold marker and setpoint marker. Each numerical display item can be programmed to display any channel value as required. The 13 digit alphanumeric display area has 20 lines available to scroll through and may be used to display any channel value or to display warning messages.

Lap times may be displayed when connected to a MoTeC Lap Beacon (or a driver activated switch). Other performance information may be displayed, including

minimum corner and maximum straight speed, fuel used or fuel remaining, and many more.

.1.1.1.6 Alarms

Warning alarms may be defined for any analog, digital, serial or calculated channel. Alarm limits are fully programmable and may include up to 6 conditions to ensure that the alarms are only activated at the correct time.

When an alarm condition has been detected, a message can be shown on the display and an auxiliary output activated. These outputs can be used for warning lights or the control of other devices.

The alarms remain active until they are acknowledged, either by activating a switch or automatically after a definable period of time.

.1.1.1.7

Controller Area Network (CAN)

The CAN bus is a high speed communication standard operating at speeds up to 1Mbit.

CAN allows many devices to be connected by a common bus, allowing all devices to share information as part of a larger system.

CAN devices include: automatic transmission controllers, sensors, multi-channel input/output modules, engine management systems etc.

.1.1.1.8 Host Software

The ADL is supplied with software packages for managing the ADL, analysing the logged data and monitoring a telemetry link.

Ease of use is one of the most attractive aspects of the MoTeC ADL software. There is no complex language to learn, just simple menu driven windows.

A full online help system is easily accessible and is integrated throughout the software.

.1.1.1.9 Dash Manager Software

The Dash Manager Software is used to configure the ADL and download logged data. It is logically laid out, giving the user access to the power of the ADL without requiring high levels of computer knowledge or intense training.

.1.1.1.10 Interpreter Software

The Interpreter software contains predefined configurations for easy data analysis. Screen display formats may be varied to suit all preferences, including user defined graphs, histograms and statistical summaries. By utilising these different display methods, users can view data in many formats to obtain accurate, meaningful analysis.

Data can also be exported into ASCII CSV file format for analysis in other software packages.

The Pro Logging option includes graph overlays, virtual instruments, mathematical functions, XY graphs (scatter plots), track maps (shows minimum and maximum speeds, gear change point and breaking points) and other advanced features.

.1.1.1.11 Telemetry Monitor (Optional)

The Telemetry Monitor software allows for realtime viewing of the telemetry data either via direct serial communications, modems or radio modems. Data can be viewed in various formats such as charts, bar graphs, dial gauges, numerics, lights, XY graphs and scroll charts. All objects are definable by the user.

.1.1.1.12 Upgrades and Accessories

The MoTeC ADL is completely field updateable by the user. The control software and logged data is stored in FLASH memory. No programming interface is required, simply send to the ADL the new program and the latest features are immediately available.

.1.1.1.13 Upgrade Options

The ADL has field upgradeable options using a password enabling system. Upgrade options include:

Extended inputs & Outputs, Pro Logging (advanced data analysis), Medium Logging (1Mbyte), Large Logging (4Mbyte), Telemetry Support, Remote Logging and Wideband Lambda measurement.

Three wiring options are available for the ADL:

Separate I/O Terminal Module with plug-in screw terminals. Includes a Realtime Clock, additional RS232 port and wide ranging power supply.

Standard (vehicle style) wiring loom for specific permanent installations.

Custom wiring looms for complex installations.

.1.1.1.14 Accessories

A wide range of sensors are available for use with the ADL including: linear position, accelerometers, pressure, resistive and thermocouple temperature sensors, hall and magnetic speed sensors and many others.

The MoTeC Lap Beacon transmitter and receiver has been designed in conjunction with the ADL. It features high channel count (990), improved optics, low power consumption and multi beacon capability.

And for peace of mind the MoTeC ADL offers a full 2 year worldwide warranty.

G-2: Advanced dash logger - specifications

ADL Specifications

.1.1.1.1 General

- Microprocessor: 32 Bit High Performance
- Manufacturing Quality standard to ISO9001
- Field updateable Operating System
- Non-volatile FLASH memory for data & operating system
- High RFI Immunity
- Rugged Aluminium Housing (IP-55, NEMA 4)
- 79 pin Autosport connector
- Operating Temperature: -10 to 70 DegC
- Operating Voltage: 7 to 22 VDC
- Operating Current: 0.3 A max.
- Weight: 385 gms (0.85 lbs)
- Size: 180mm x 91mm x 18mm (excluding connector)
- Reverse Battery and Transient Protection
- Warranty: 2 years Parts and Labour

.1.1.1.2 Measurement Inputs

- 28 Analog Inputs (10 Standard):
  - 20 Analog Voltage (6 Standard)
  - 8 Analog Temperature (4 Standard)
  - 12 bit resolution, 0 to 15 VDC range
  - Update rate (Max. 8 channels): up to 1000 times/sec
  - Other inputs: up to 500 times/sec

- 4 Digital Inputs (2 Standard)
- 4 Speed Inputs (2 Standard)

  Digital & Speed
  - Switch to OV, logic signal, open collector (Hall Effect), or Magnetic
  - State & Counting (1MHz)
  - Period (1 micro sec)
  - Pulse width (1 micro sec)

- 4 Switch Inputs (4 Standard
- User definable sensor calibrations

.1.1.1.3 Auxiliary Outputs

- 8 Digital Outputs (4 standard)
  - Open Collector (drives to ground) with pullup (10k ohms) to battery positive
  - On/Off or Pulse Width Modulation with variable Frequency and Duty Cycle

.1.1.1.4 Air Fuel Ratio Measurement (Optional)

- 2 high accuracy Wideband Lambda (Air/Fuel ratio) inputs
- Resolution: 0.01 Lambda

- Temperature compensated
- Range: 0.75 to 1.2 Lambda

.1.1.1.5 Data Logging

- Memory: 384k, 1MB, 2MB, 4MB, 8MB
- Non-volatile FLASH, field upgradeable
- Logging of any Analog, Digital, Serial, CAN bus or Calculated channel
- Maximum Logging throughput: 20k/sec
- 2 Burst Logging buffers with pre triggering (Large logging option only)
- Typical Unload Speed: 19 sec/MB, using parallel port of PC to CAN bus RS232 unload rates dependent on baud rate

.1.1.1.6 Calculations

- Timers (0.01s, 0.1s, & 1s resolution)
- 2D and 3D Tables
- User conditions
- Math Functions: Differentiate, Integrate, Absolute, Min/Max
- Lap Time and Number
- Lap Gain/Loss
- Speed and Distance
- Gear Detection
- Fuel Prediction
- Tell-tales
- Running Min/Max

.1.1.1.7 Display

- Custom LCD, High Contrast, High Temperature, Reflective
- Display any Analog, Digital, Serial, CAN bus channel or Calculated channel
- 3 Display Modes
- 70 Segment Bar graph
    - Definable Range
    - Programmable Setpoint and Peak Hold point

- 4 Numeric Display Items
- 13 Digit Alpha Numeric Display area, 1,2 or 3 channels per line (20 scrollable lines per display mode)
    - Alarm messages
    - Channel display
    - Descriptive text

.1.1.1.8 Communication

- Serial RS232 Coms. (1200 to 115k baud)
- CAN data link (250Kbit to 1Mbit)
- Telemetry Link output (RS232)

.1.1.1.9 Host Software

1. Dash Manager Software
2. Interpreter Analysis Software
3. Telemetry Software (Optional)

Computer Requirements

- IBM PC compatible running Windows 95/98 or NT4.0
- Pentium (Min.) 90MHz, 16MB RAM

.1.1.1.10 Upgrades

The MoTeC ADL in its base configuration includes:

- 10 Analog Inputs
- 8 Digital Inputs
- 4 Digital Auxiliary Outputs
- RS232 and CAN bus support
- Software: Dash Manager and Interpreter
- User's Manual

Upgrades Available *(field updateable by the user)*:

- **Extended Inputs & Outputs**
  - 28 Analog Inputs (10 standard)
  - 12 Digital Inputs (8 standard)
  - 8 Digital Auxiliary Outputs (4 standard)

- **Pro Logging** - Advanced Analysis Software
  - Graph Overlays
  - XY Plots
  - Maths functions
  - Virtual Instruments display
  - Track Mapping

- **Medium Logging**
  - 384k to 1MB Memory

- **Large Logging** *(requires Medium Logging Upgrade)*
  - 1MB to 8MB with Burst mode logging

- **Lambda Measurement**
  - 2 Wideband Lambda inputs

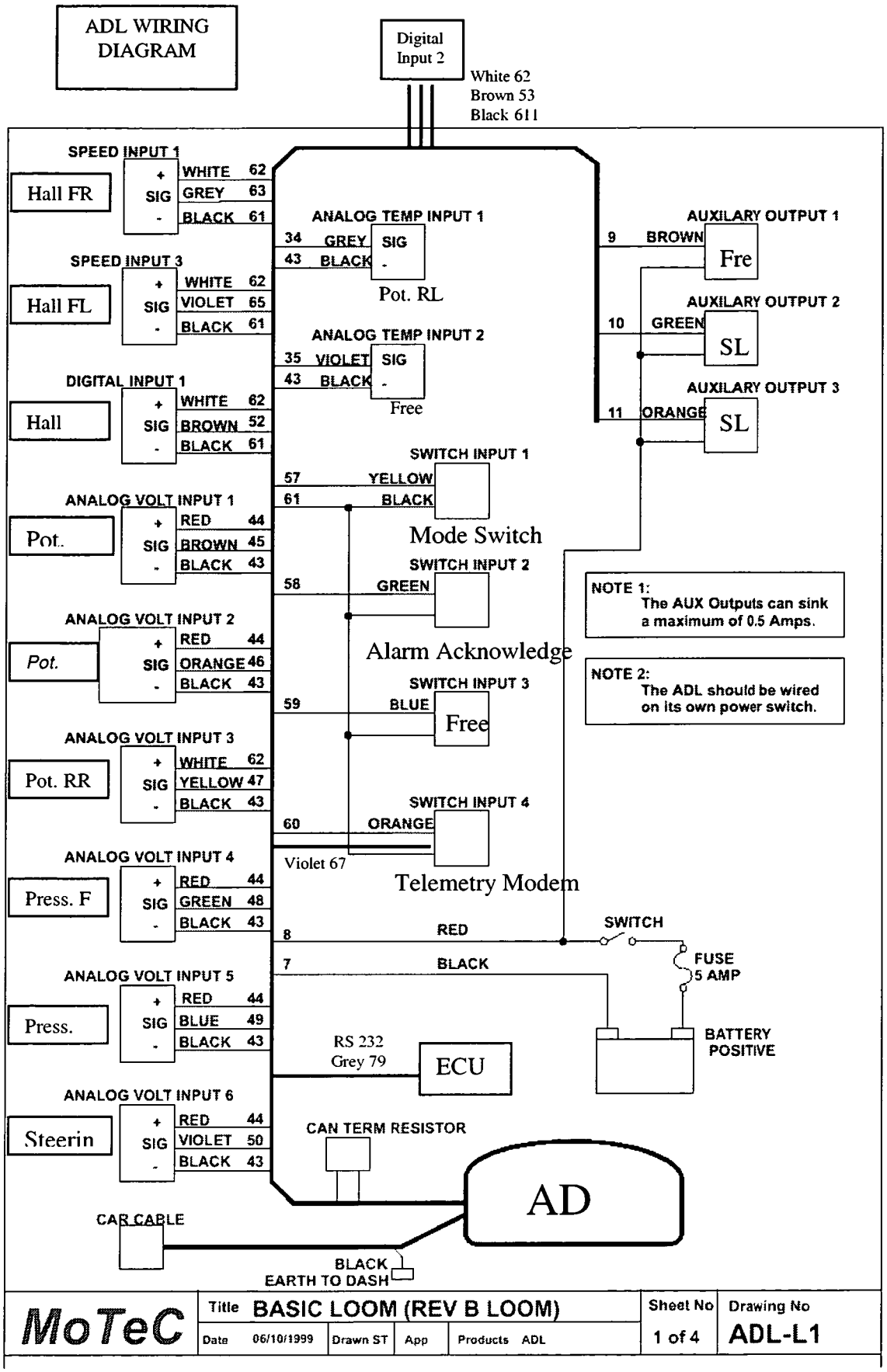
- **Telemetry**
  - Enables realtime viewing of data via a telemetry link

- **Remote Logging** *(requires Telemetry Upgrade)*
  - Allows Remote Logging via a telemetry link or hand held computer

.1.1.1.11 Accessories

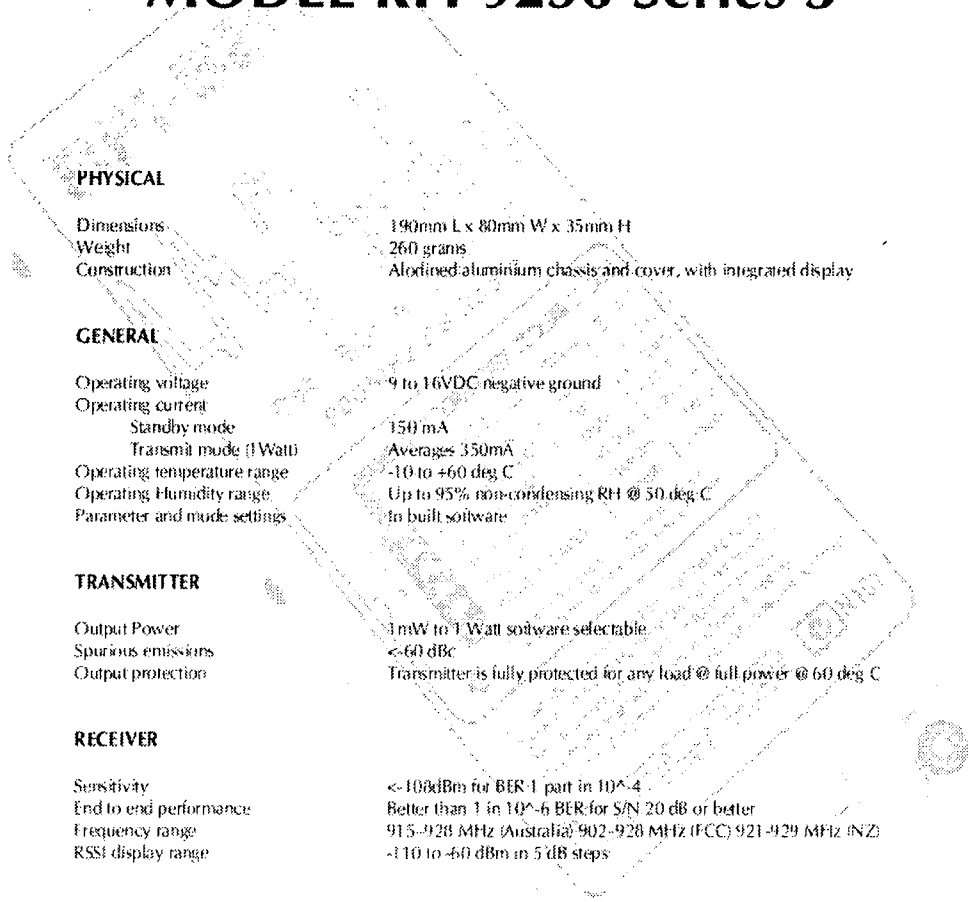
- PC Communications Cable (High Speed CAN)
- Wiring Looms
- Input/Output Terminal Module
- Lambda (Air/Fuel ratio) Sensors and Kits
- Telemetry Products
  - GSM mobile phones, radio modems etc
- Sensors and transducers
  - a full range of sensors, amplifiers, transducers, lights and buttons are available
- Lap Beacon: Transmitter and Receiver (990 channel)

G-3: ADL wiring loom



G-4: Radio modem – specifications

Specifications
MODEL RFI 9256 Series 3



PHYSICAL

Dimensions	190mm L x 80mm W x 35mm H
Weight	260 grams
Construction	Alodined aluminium chassis and cover, with integrated display

GENERAL

Operating voltage	9 to 16VDC negative ground
Operating current	
Standby mode	150 mA
Transmit mode (1 Watt)	Averages 350mA
Operating temperature range	-10 to +60 deg C
Operating Humidity range	Up to 95% non-condensing RH @ 50 deg C
Parameter and mode settings	In built software

TRANSMITTER

Output Power	1mW to 1 Watt software selectable
Spurious emissions	<-60 dBc
Output protection	Transmitter is fully protected for any load @ full power @ 60 deg C

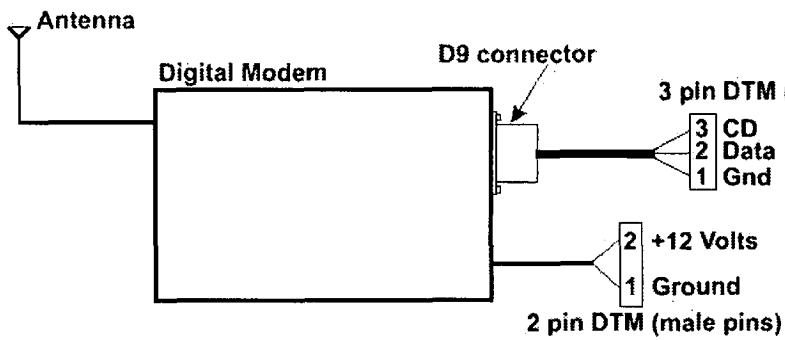
RECEIVER

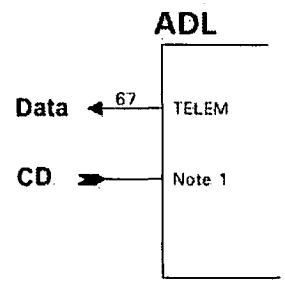
Sensitivity	<-108dBm for BER 1 part in 10^-4
End to end performance	Better than 1 in 10^-6 BER for S/N 20 dB or better
Frequency range	915-928 MHz (Australia) 902-928 MHz (FCC) 921-929 MHz (NZ)
RSSI display range	-110 to -60 dBm in 5 dB steps

DATA SYSTEM

RS 232 handshaking	Hardware/ software/ none software selectable
Protocol modes	All common variants / PLCs supported including point / point, point / multipoint and Hayes modes
Interface data speed	110 to 115,000 bps, software selectable all modes

Distributed by:

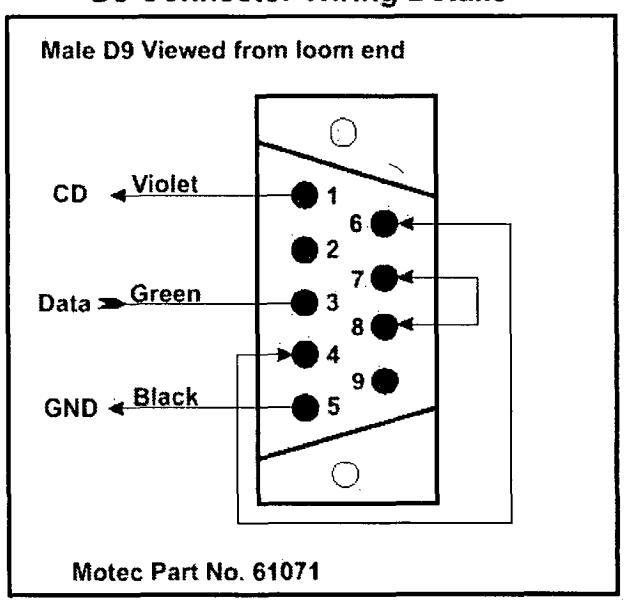




Note 1

The following pins can be used:
Switched Inputs (normally sw4)
Speed Inputs, Digital Inputs

D9 Connector Wiring Details



ADL Setup

1. Set Input pin's channel to "Carrier Detect"
2. Select one of the templates in the RS232 Communications Setup screen.

NOTE: The modem may need configuration before it will connect.

MoTeC	Title: Digital Telemetry Connection				Sheet No	Drawing No
	Date 04/03/1999	Drawn ST	App	Products ADL	1 of 1	TEL01

G-6: Wheel speed hall effect sensor – specifications

Solid State Sensors

GT1 Series

Hall Effect Gear Tooth Sensors

SENSOR SPECIFICATIONS

All values were measured using 1 K pull-up resistor.

Electrical Characteristics	Supply Voltage	4.5 to 24 VDC
	Supply Current	10 mA typ., 20 mA max.
	Output Voltage (output low)	0.4 V max.
	Output Current (output high)	10 µA max. leakage into sensor
	Switching Time	
	Rise (10 to 90%)	15 µsec. max.
	Fall (90 to 10%)	1.0 µsec. max.
Absolute Maximum Ratings*	Supply Voltage (Vs)	±30 VDC continuous
	Voltage Externally Applied To Output (output high)	−0.5 to +30 V
	Output Current	40 mA sinking
	Temperature Range	
	Storage	−40 to 150° C (−40 to 302°F)
	Operating	−40 to 150° C (−40 to 302°F)
Switching Characteristics**	Operate Point	3.7±1.25° (3,28±1.13 mm)
	Release Point	4.7±2.50° (4,16±2,21 mm)
	Differential Travel	8.4±3.70° (7.45±3.34 mm)

* As with all solid state components, sensor performance can be expected to deteriorate as rating limits are approached; however, sensors will not be damaged unless the limits are exceeded.
** See Reference Target table.

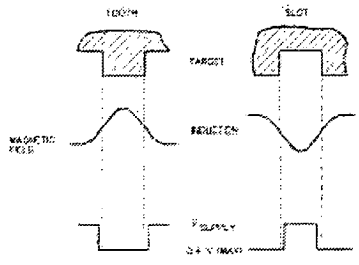
TARGET GUIDELINES

The Target Guidelines table provides basic parameters when an application is not restricted to a specific target.

Any target wheel that exceeds the following minimum specifications can be sensed over the entire temperature range of −40° to 150°C with any sensing gap up to .080 in. (2.0 mm). This data is based on a 4 in. (102 mm) diameter wheel, **rotating 10 to 3600 RPM.**

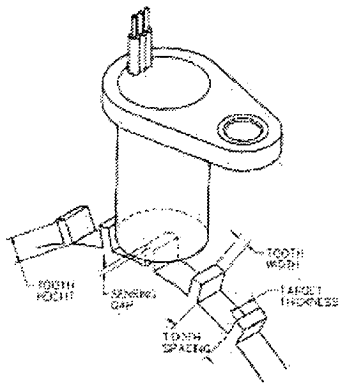
Reference Target Dimensions	
Tooth Height:	.200 in. (5.08 mm) min.
Tooth Width:	.100 in. (2.54 mm) min.
Tooth Spacing:	.400 in. (10.16 mm) min.
Target Thickness:	.250 in. (6.35 mm)

Sensor Output (with pull-up resistor added to output circuit)



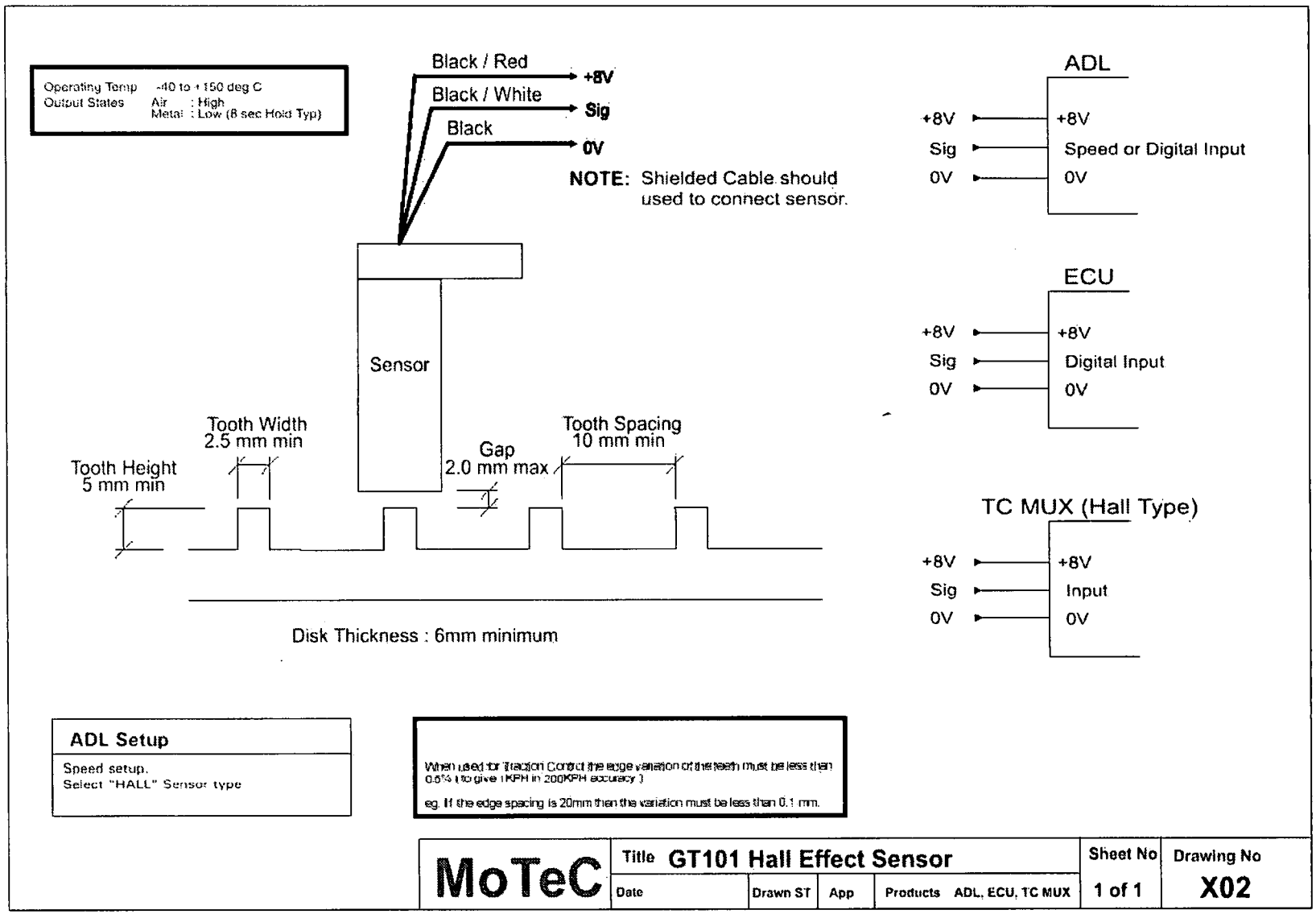
REFERENCE TARGET/CONDITIONS

Characteristics will vary due to target size, geometry, location, and material. Sensor specifications were derived using a cold-rolled steel reference target. See table, right, for reference target configuration and evaluation conditions.



Target	
Diameter:	4 in. (101,6 mm)
Tooth Width:	.350 in. (8,89 mm)
Thickness:	.250 in. (6,35 mm)

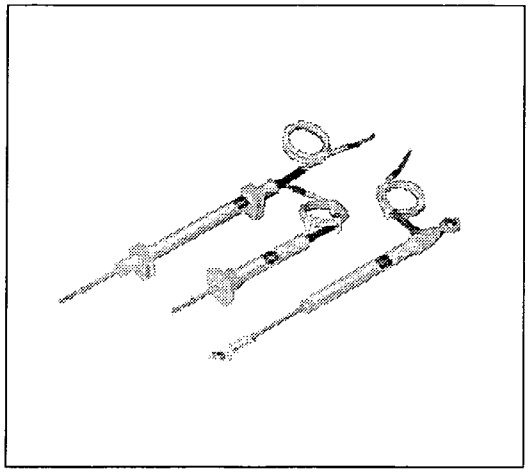
Test Conditions	
Air Gap:	.040 to .080 in. (1,02 to 2,03 mm)
V Supply:	4.5 to 24 V
RPM:	10 min., 3600 max.


Operating Temp -40 to +150 deg C
Output States Air : High
Metal : Low (8 sec Hold Typ)
Black / Red
+8V
Black / White
Sig
Black
0V
NOTE: Shielded Cable should used to connect sensor.
Sensor
Tooth Width 2.5 mm min
Tooth Height 5 mm min
Gap 2.0 mm max
Tooth Spacing 10 mm min
Disk Thickness : 6mm minimum
ADL
+8V
Sig
0V
+8V
Speed or Digital Input
0V
ECU
+8V
Sig
0V
+8V
Digital Input
0V
TC MUX (Hall Type)
+8V
Sig
0V
+8V
Input
0V
ADL Setup
Speed setup.
Select "HALL" Sensor type
When used for Traction Control the edge variation of the teeth must be less than 0.5% (to give 1KPH in 200KPH accuracy)
eg. If the edge spacing is 20mm then the variation must be less than 0.1 mm.
MoTeC
Title GT101 Hall Effect Sensor
Date
Drawn ST
App
Products ADL, ECU, TC MUX
Sheet No 1 of 1
Drawing No X02

G-8: Suspension position linear potentiometer – specifications

GEFRAN

PZ12
RECTILINEAR DISPLACEMENT TRANSDUCER WITH CYLINDRICAL CASE



Main features

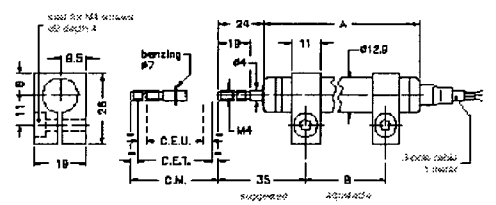
- *25 to 150 mm. stroke*
- *Mechanical fixing using brackets, selfaligning ball-joints or flange*
- *Independent linearity up to ± 0,05%*
- *Infinite resolution*
- *No variation of electrical signal outside theorical electrical stroke*
- *Displacement speed up to 10 m/s*
- *Working temperature: -30...+100°C*
- *Electrical connection: 3-pole screened cable (1m length)*
- *Life duration: > 25x10⁶ meters or >100x10⁶ operations whichever is the smaller (within C.E.U.)*
- *Grade of protection IP60*

TECHNICAL DATA

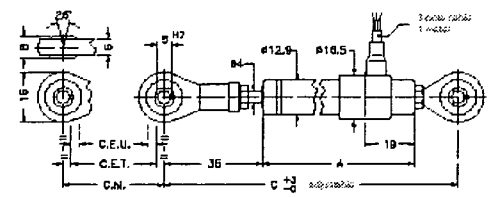
Useful electrical stroke (C.E.U.)	25/50/75/100/125/150
Independent linearity (within C.E.U.)	see table
Displacement speed	≤ 10 m/s
Displacement force	≤ 0.5N
Vibrations	5...2000Hz, Amax =0,75 mm amax. = 20 g
Shock	50 g, 11ms.
Tolerance on resistance	± 20%
Recommended cursor current	< 0,1 µA
Maximum cursor current	10mA
Maximum applicable voltage	see table
Electrical isolation	>100MΩ at 500V=, 1bar, 2s
Dielectric strength	< 100 µA at 500V~, 50Hz, 2s. 1bar
Dissipation at 40°C (0W at 120°C)	see table
Temp. Coefl. of the resistance	-200 ± 200ppm/°C
Actual Temperature Coefficient of the output voltage	< 1.5ppm/°C
Working temperature	-30...+100°C
Storage temperature	-50...+120°C
Case material	Anodised aluminium Nylon 66 GV 40
Control rod material	Stainless steel AISI 303
Fixing	Brackets, selfaligning ball-joints or flange

MECHANICAL DIMENSIONS

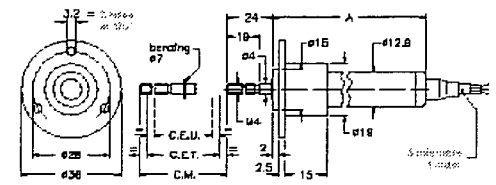
PZ12-S



PZ12-A



PZ12-F

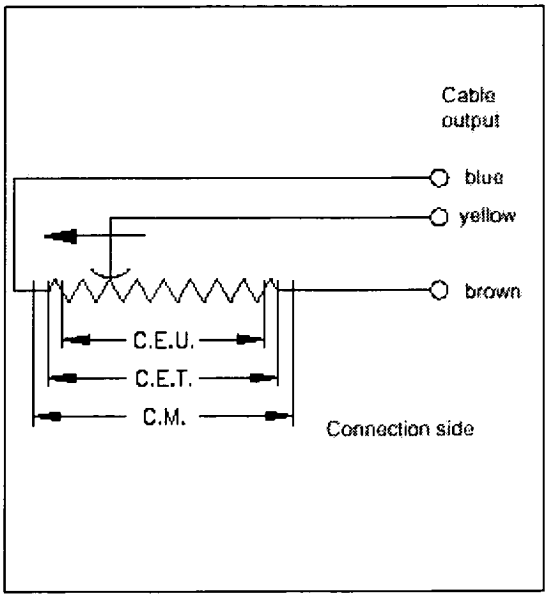


Important: all the data reported in the catalogue linearity, lifetime, temperature coefficient are valid for a sensor utilization as a ratiometric device with a max. current across the cursor lc ≤ 0.1 µA.

MECHANICAL / ELECTRICAL DATA

MODEL			25	50	75	100	125	150
Useful electrical stroke (C.E.U.) + 1 / -0		mm	25	50	75	100	125	150
Theoretical electrical stroke (C.E.T.) ± 1		mm	C.E.U. +1					
Resistance (C.E.T.)		kΩ	1	2	3	4	5	6
Independent linearity (within C.E.U.)		± %	0,2	0,1	0,1	0,1	0,05	0,05
Dissipation at 40°C (0W at 120°C)		W	0,5	1	1,5	2	2,5	3
Maximum applicable voltage		V	20	40	60			
Mechanical stroke (C.M.)		mm	C.E.U. +5					
Case length (A)	mod. PZ12 - S	mm	74,5	99,5	124,5	149,5	174,5	199,5
	mod. PZ12 - A	mm	102	127	152	177	202	227
	mod. PZ12 - F	mm	74,5	99,5	124,5	149,5	174,5	199,5
Recommended distance between brackets (B)		mm	42	67	92	117	142	167
Minimum distance between ball-joints (C)		mm	153	178	203	228	253	278
Weight	mod. PZ12 - S	g	45	55	65	75	85	95
	mod. PZ12 - A	g	70	80	90	100	110	120
	mod. PZ12 - F	g	60	70	80	90	100	110

ELECTRICAL CONNECTIONS



STANDARD ACCESSORIES

	Code
2 mounting brackets for PZ12-S	STA074

ORDER CODE

Displacement transducer **PZ12** □ □ □

Mounting by brackets	S
Mounting by self-aligning ball-joints	A
Mounting by flange	F

Model

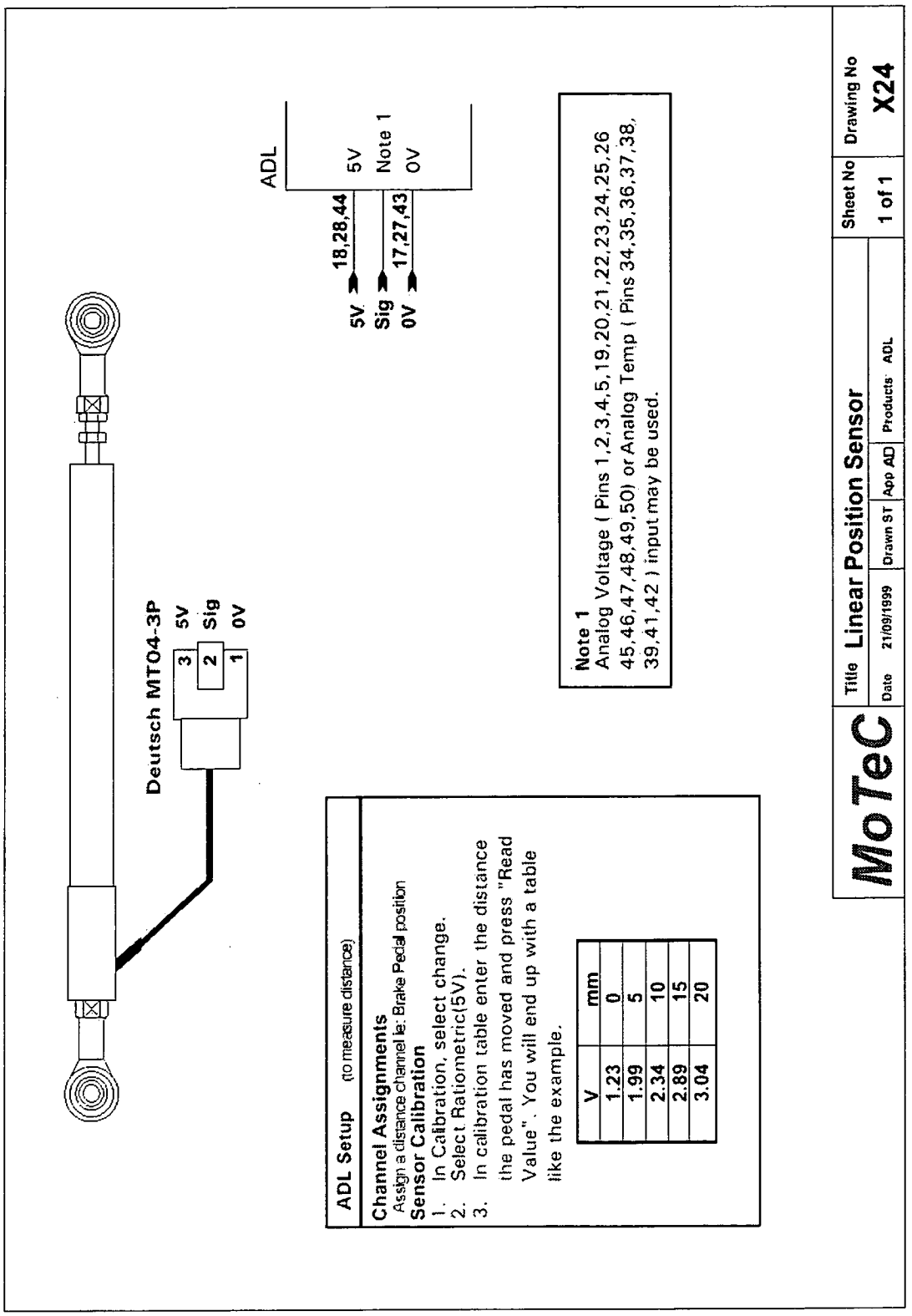
If requested, it is possible to supply models with non-standard mechanical and/or electrical features.

Example: **PZ12 - S - 25**
Displacement transducer model PZ12, mounting by brackets, useful electrical stroke (C.E.U.) 25mm

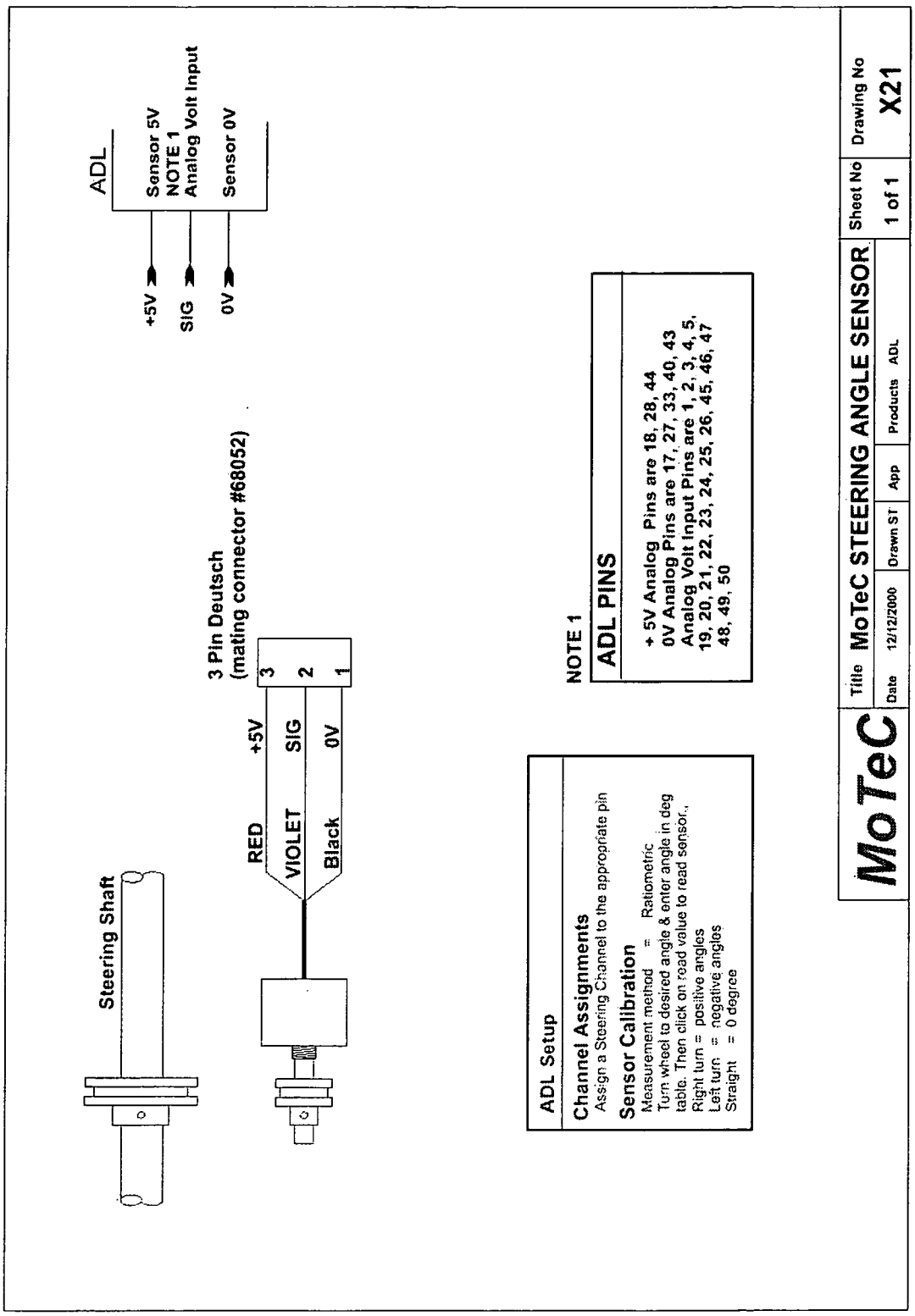
GEFRAN spa
via Sebina, 74
25050 PROVAGLIO D'ISEO (BS) - ITALIA
ph. 0309888.1 - fax. 0309839063
Internet: http://www.gefran.com

cod. 84875-10/99

G-9: Suspension position linear potentiometer – installation



G-10: Steering angle radial potentiometer – installation



G-11: Brake pressure transducer - specifications

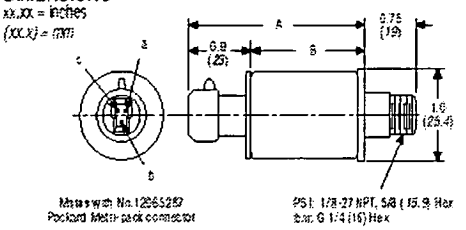
ECLIPSE® • *OEM PRESSURE TRANSDUCER*

TECHNICAL SPECIFICATIONS

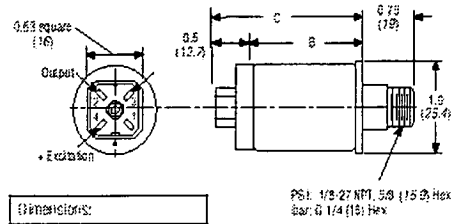
RANGE		
	0-15, 25, 50 PSIG 0-100, 250, 500, 1000, 2000, 3000, 5000, 7100 PSIS 0-1, 2, 4, 7 bar g (0-10, 15, 20, 35, 50, 100, 200, 350, 700 bar s)	
PHYSICAL		
Proof Pressure	1.5 x rated range	
Burst Pressure	5 x rated range	
Material in Contact With Media	300 series stainless steel, braze compound	
Shock	50 g's peak (5 milliseconds)	
Vibration	Meets MIL-STD-810-C, Figure 514.2-5, Curve AK, 20.7 g rms minimum	
ELECTRICAL	**Voltage output**	**Current output**
Full Scale Output	4.00 Vdc nominal (0.5 - 4.5 Vdc)	16 mA into 0 to 1400 loop resistance (4-20 mA)
Zero Output	0.5 V nominal	4 mA nominal
Excitation	5.0 Vdc ± 0.25 Vdc @ 20mA	10 to 40 Vdc (linear derating to 35 Vdc from 25°C to 100 C°)
Reverse Polarity Protection	Yes	
Insulation Resistance	1000 M @ 50 Vdc	
Electrical Connection	Standard Packard Metri-Pack™ Requires Packard #12065287 mating connector, not included. Optional Hirschmann connector, mate included.	
PERFORMANCE		
Accuracy	±1% of FSO from best fit straight line including effects of nonlinearity, hysteresis and nonrepeatability	
Operating Temperature Range	-40° to 105°C (-40° to 221°F)	
Compensated Temperature Range	-1° to 82°C (30° to 180°F)	
Total Error	±4% of full scale. Includes the effects of zero output error, calibration error, temperature, nonlinearity, hysteresis, and repeatability.	

DIMENSIONS
xx.xx = inches
(xx.x) = mm

Eclipse with Packard

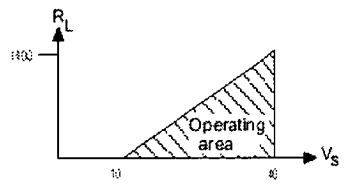


Eclipse with Hirschmann

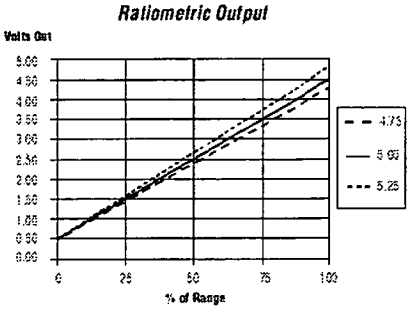


Dimensions:		
	Voltage	**Current**
A	2.4 (61)	2.6 (66)
B	1.5 (38)	1.7 (43)
C	2.0 (51)	2.2 (56)

Load resistance in current loop



Ratiometric Output

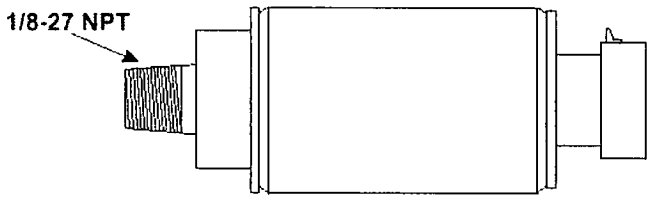


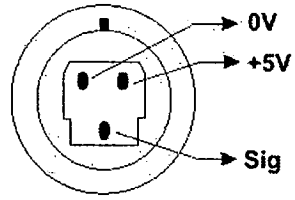
PACKARD CONNECTOR PINS

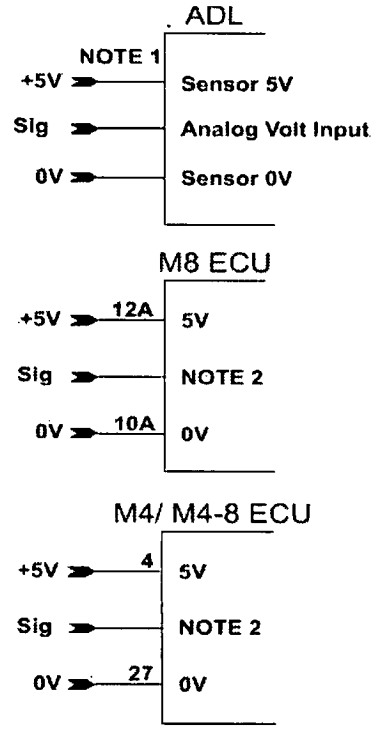
	Voltage	**Current**
a	+ Excitation	+ Excitation
b	Output	- Excitation (Return)
c	Common	NC

HIRSCHMANN CONNECTOR PINS

	Voltage	**Current**
Pin 1	NC	NC
Pin 2	Signal Output	NC
Pin 3	Common	- Excitation (Return)
Pin 4	+ Excitation	+ Excitation







ADL Setup

Channel Assignments
Assign the pressure Channel to the appropriate pin

Sensor Calibration
Press "Select" and choose correct calibration file for the sensor
For example, Data Inst Eclipse 100 psi MAP.C1P

NOTE 1:
+ 5V Analog Pins are 18, 28, 44
0V Analog Pins are 17, 27, 33, 40, 43
Analog Volt Input Pins are 1, 2, 3, 4, 5, 19, 20, 21, 22, 23, 24, 25, 26, 45, 46, 47 48, 49, 50

NOTE 2:
M8 input pins are 28A,27A,13B,7B,8B,9B 14B, 15B & 16B
M4 and M4-8 input pins are 17,18 & 30

ECU Setup M8	
Sensor Calibration	100 PSI sensor
Display in KPA	17
Display in PSI	23
Display in InHg	29

ECU Setup M4 M4-8	
Sensor Calibration	100 PSI sensor
MAP PIN	-1*
AUX TEMP PIN	-4*
AUX VOLT PIN	-3*

* = DISPLAY IN KPA

100 PSI Sensor CalibrationTables for M4-M4-8

Map Pin	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250
	102	114	125	137	149	161	172	184	196	208	219	231	243	254	266	278	290	301	313	325	337	348	360	372	384	395
	260	270	280	290	300	310	320	330	340	350	360	370	380	390	400	410	420	430	440	450	460	470	480	490	500	
	407	419	430	442	454	466	477	489	501	513	524	536	548	559	571	583	598	606	618	630	642	653	665	677	689	

Aux Temp & Aux VoltPin	0	40	80	120	160	200	240	280	320	360	400	440	480	520	560	600	640	680	720	760	800	840	880	920	960	1000
	102	149	196	243	290	336	383	430	477	524	571	618	665	711	758	805	852	899	940	981	1018	1054	1091	1127	1164	1200

MoTeC	Title DATA INSTRUMENTS (ECLIPSE)				Sheet No	Drawing No
	Date 28/07/1999	Drawn ST	App AD	Products ADL, ECU	1 of 1	X22

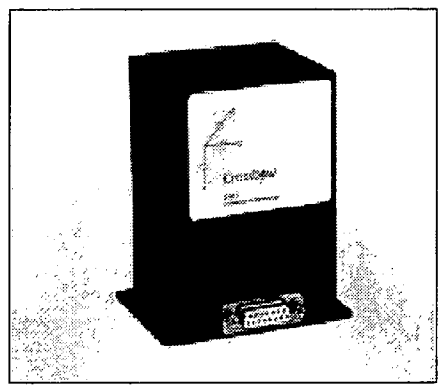
G-13: Attitude & heading reference sensor – specifications and installation

New

AHRS

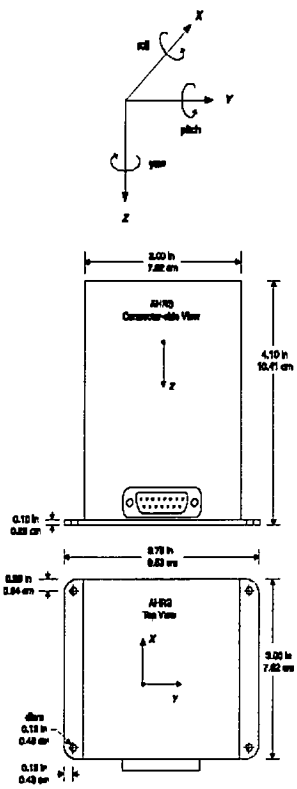
ATTITUDE & HEADING REFERENCE SYSTEM

- Roll, Pitch and Heading Angle in Dynamic Environments
- Enhanced Performance Kalman Filter Algorithm
- High Stability MEMS Sensors
- High Range Gyro and Accel Options



Applications

- UAVRPV Control
- Platform Stabilization
- Avionics



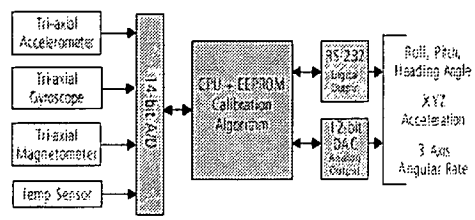
AHRS400CA (DMU-HDX-AHRS)

The Crossbow AHRS400CA is a high performance, solid-state attitude and heading reference system intended for airborne applications such as UAV control, Avionics, and Platform Stabilization. This high reliability, strap-down inertial subsystem provides attitude and heading measurement with static and dynamic accuracy comparable to traditional spinning mass vertical and directional gyros.

This AHRS400 series product builds on the performance of the AHRS300 series. It features higher performance sensors, including silicon MEMS accelerometers and gyroscopes with lower noise and improved bias stability.

The AHRS400CA achieves its excellent performance by employing proprietary Kalman Filter algorithms to determine stabilized roll, pitch, and heading angles in static and dynamics conditions. The Kalman Filter implementation results in a continuous on-line gyro bias calibration, and an adaptive attitude and heading measurement that is stabilized by the long term gravity and magnetic north references. Output data is provided in both analog and digital (RS-232) formats.

Each Inertial System comes with a User's Manual offering helpful hints on programming, installation, and product information. In addition, Crossbow's GYRO-VIEW software is included to assist you in system development and evaluation, and allows you to perform data acquisition.



AHRS Block Diagram

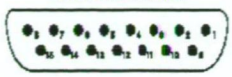
inertial systems

Crossbow

Specifications		AHRS400CA-100	AHRS400CA-200	Remarks
Performance				
Update Rate (Hz)		>60	>60	Continuous update mode
Start-up Time Valid Data (sec)		<1	<1	
Fully Stabilized Data (sec)		<60	<60	
Heading				
Range (°)		±180	±180	
Static Accuracy (°)		<±1.5	<±2	
Dynamic Accuracy (° rms)		±3	±4	
Resolution (° rms)		<0.1	<0.1	
Attitude				
Range: Roll, Pitch (°)		±180, ±90	±180, ±90	
Static Accuracy (°)		<±0.5	<±1	Measured on level surface
Dynamic Accuracy (° rms)		±2.0	±2.5	
Resolution (°)		<0.1	<0.1	
Angular Rate				
Range: Roll, Pitch, Yaw (°/sec)		±100	±200	
Bias: Roll, Pitch, Yaw (°/sec)		<±1.0	<±1.0	Scaled sensor mode
Bias: Roll, Pitch, Yaw (°/sec)		<±0.05	<±0.05	Angle mode
Scale Factor Accuracy (%)		<1	<1	
Non-Linearity (% FS)		<0.3	<0.3	
Resolution (°/sec)		<0.025	<0.05	
Bandwidth (Hz)		>10	>10	-3 dB point
Random Walk (°/hr$^{1/2}$)		<0.85	<1.7	
Acceleration				
Input Range: X/Y/Z (g)		±2	±10	
Bias: X/Y/Z (mg)		<±8.5	<±12	
Scale Factor Accuracy (%)		<1	<1	
Non-Linearity (% FS)		<1	<1	
Resolution (mg)		<0.25	<1.25	
Bandwidth (Hz)		>10	>10	-3 dB point
Random Walk (m/s/hr$^{1/2}$)		<0.1	<0.5	
Environment				
Operating Temperature (°C)		-40 to +71	-40 to +71	
Non-Operating Temperature (°C)		-55 to +85	-55 to +85	
Non-Operating Vibration (g rms)		6	6	20 Hz - 2 KHz random
Non-Operating Shock (g)		1000	1000	1 ms half sine wave
Electrical				
Input Voltage (VDC)		9 to 30	9 to 30	
Input Current (mA)		<300	<300	
Power Consumption (W)		<4	<4	at 12 VDC
Digital Output Format		RS-232	RS-232	"See Digital Data Format"
Analog[1] Range (VDC)		±4.096	±4.096	Pins 8, 9, 10, 12, 13, 14
		0 to 5.0	0 to 5.0	Pins 5, 6, 7
Physical				
Size	(in)	3.0 x 3.75 x 4.10	3.0 x 3.75 x 4.10	Includ. mounting flanges
	(cm)	7.62 x 9.53 x 10.41	7.62 x 9.53 x 10.41	Includ. mounting flanges
Weight	(lbs)	<1.4	<1.4	
	(kg)	<0.64	<0.64	
Connector		15 pin sub-miniature "D" female		

Notes
[1]All analog outputs are fully buffered and are designed to interface directly to data acquisition equipment.
Specifications subject to change without notice

15 Pin "D" Connector Socket Pinout



Pin	Signal
1	RS-232 Transmit Data
2	RS-232 Receive Data
3	Input Power
4	Ground
5	X-axis accel voltage[3]
6	Y-axis accel voltage[3]
7	Z-axis accel voltage[3]
8	Roll-axis angular rate[2]
9	Pitch-axis angular rate[2]
10	Yaw-axis angular rate[2]
11	NC - Factory use only
12	Roll angle/X-axis mag voltage[3]
13	Pitch angle/Y-axis mag voltage[3]
14	Not used/Z-axis mag voltage[3]
15	NC - Factory use only

Notes
1. The accelerometer voltage outputs are taken directly from the accelerometers without compensation or scaling.
2. The angular rate analog outputs are scaled to represent degrees/second. Outputs are created by a D/A converter.
3. Actual output depends on AHRS measurement mode. Axis magnitude is scaled analog voltage vs. angle.

Pin Diagram

Ordering Information

Model	Previous Model	Description	Gyro (°/sec)	Accel (g)
AHRS400CA-100	DMU-HDX-AHRS	Attitude & Heading Reference System	± 100	± 2
AHRS400CA-200	DMU-HDX-AHRS	Attitude & Heading Reference System	± 200	± 10

CALL FACTORY FOR OTHER CONFIGURATIONS

inertial systems